HPL To Fluke BASIC

Conversion Manual

FLUKE®

Price $5.00

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_10600

HPL To Fluke BASIC

Conversion Manual

P/N 546341
February 1980

FLUKE®

REVISED

SEE YELLOW INSERT

All data, documentation, dialog, diagrams, suggestions, reports and/or other forms of media contained in this bulletin are intended to be informational in nature only. Implementation of such data to a user's application should ONLY be made after careful analysis by the user's own software experts. JOHN FLUKE MFG. CO., INC. SPECIFICALLY DISCLAIMS ALL WARRANTIES INFORMATION, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANTY OF MERCHANTABILITY, FITNESS, OR ADEQUACY FOR ANY PARTICULAR PURPOSE OR USE.

John Fluke Mfg. Co., Inc., • P.O. Box 43210 • Mountlake Terrace, Washington 98043

© 1980, John Fluke Mfg. Co., Inc., all rights reserved

Table of Contents

SECTION		TITLE	PAGE
1	**INTRODUCTION**		1
	1.1	Preface	1
	1.2	How to Use This Manual	2
2	**GENERAL DESCRIPTION OF FLUKE 1720A**		4
	2.1	1720A Enhancements (features Not Available on the 9825A)	5
	2.2	Power Up Procedure	8
	2.3	Access to BASIC Interpreter Mode	9
	2.4	System Software Mode Configuration	9
	2.4.1	Command File	10
	2.4.2	Console Monitor	11
	2.4.3	File Utility Program	12
3	**1720A-9825A PROGRAMMING COMPARISON**		15
	3.1	Equivalent Programming Capabilities	15
	3.2	Essential Programming Differences	16
	3.3	Programming Enhancements	18
	3.4	Rules for Translation	20
	3.5	Special Key Equivalents	22
	3.5.1	Control Keys	22
	3.5.2	Special Function Keys	23
	3.5.3	System Command Keys	24
	3.5.4	Line Editing Keys	25
	3.5.5	Character Editing Keys	25
	3.5.6	Display Keys	26
	3.5.7	Miscellaneous Important Keys	26
	3.6	Operator Equivalents	27
	3.7	User Assigned Variable Equivalents	29
	3.8	Flag Equivalents	33
	3.9	Special Flags	34
	3.10	Command Equivalents	40
	3.11	Standard Statement Equivalents	41
	3.12	Standard Function Equivalents	44
	3.13	Tape Cartridge Operation Equivalents	48
	3.14	String ROM Statement Equivalents	53
	3.15	Advanced Programming ROM Statement Equivalents	56
	3.16	General I/O ROM Statement Equivalents	61
	3.17	Flexible Disk ROM Statement Equivalents	64
	3.18	Extended I/O ROM Statement Equivalents	70
	3.19	System Programming ROM Statement Equivalents	89
4	**1720A PROGRAMMING TIPS**		90
	4.1	Programming Pitfalls	90
	4.2	Programming Enhancements	93
	4.2.1	Touch Sensitive Display	94
	4.2.2	CRT Attributes	99
	4.2.3	CRT Graphics	101
	4.3	Programming Shortcuts	104
	4.4	Obtaining a Printed Program Listing	105

SECTION	TITLE	PAGE
5	**APPENDICES**	**107**
5A	Algorithm for Multi-Dimensional Array	107
5B	Device, Channel, File and Port Designations	109
5C	Programming Conventions	110
5D	Variable Conventions	111
5E	1720A BASIC to HPL Equivalents Cross Reference	113
5F	Mathematic Algorithms	116
5F.1	Degrees vs. Radians	116
5F.2	Grads vs. Radians	117
5F.3	ArcSin and ArcCos Defined Functions	118
5G	Decimal Equivalents for IEEE-488 Commands	120
5H	ASCII Character Codes	121

List of Tables

SECTION	TABLE	TITLE	PAGE
2	2-1	1720A Enhancements	5
	2-2	Console Monitor Tasks	11
3	3-1	Key Equivalents	22
	3-2	Operator Equivalents	27
	3-3	Variable Equivalents	29
	3-3a	Dimension Equivalents	30
	3-4	Command Equivalents	40
	3-5	Statement Equivalents	41
	3-6	Function Equivalents	44
	3-7	Tape Cartridge Equivalents	48
	3-8	String Equivalents	53
	3-9	Advanced Statement Equivalents	56
	3-10	General I/O Statement Equivalents	61
	3-11	Disk Statement Equivalents	64
	3-12	Extended I/O Equivalents	70
	3-13	Systems Programming Equivalents	89
4	4-1	Attributes and Graphics	103

List of Figures

SECTION	FIGURE	TITLE	PAGE
2	2-1	1720A Controller	8
	2-2	System Software Mode Configuration	14
4	4-1	Operator Interface	94
	4-2	Operator Display	95

TABLE OF PROGRAMS

Program #	Section	Program Name	Description
1	2.4.1	FUP.HLP	Summary of Options in FUP
2	3.7	3DARAY	3-Dimensional Arrays (HPL and BASIC)
3	3.8	ERROR	Error Handler (HPL and BASIC)
4		ADD1	Adding Machine (HPL and BASIC)
5	3.12	PRND	Power Rounding (BASIC only)
6		DRND	Digit Rounding (BASIC only)
7		INTFRC	Integer/Fractional Parts Using Strings (BASIC only)
8		MAXMIN	Maximum and Minimum Value Search (BASIC only)
9	3.13	SDATA	Data Handler 1 (BASIC only)
10		VDATA	Data Handler 1A (BASIC only)
11		SPRT	Data Handler 2 (BASIC only)
12		PDATA	Data Handler 3 (HPL and BASIC)
13	3.5	FTSSTF	Full Split/Split to Full (HPL and BASIC)
14		FTIITF	Full to Integer/Integer to Full (BASIC only)
15	3.16	TRIG2	Trigonometric Table (HPL and BASIC)
16	3.17	DDATA	Data Handler 4 (BASIC only)
17		HDATA	Data Handler 5 (BASIC only)
18		VARAY1	Data Handler 6A (BASIC only)
19		VARAY2	Data Handler 6B (BASIC only)
20	3.18	5100VR	5100A Verification (HPL and BASIC)
21	4.4	LIST.13E	Obtaining a Program Listing on IEEE-488 (BASIC only)
22	4.5	TOUCH	CRT Touch Sense (BASIC only)
23	4.6	ATDEMO	CRT Attributes (BASIC only)
24		GRDEMO	CRT Graphics (BASIC only)

SECTION 1

INTRODUCTION

1.1 PREFACE

The FLUKE 1720A Controller is the first of its kind to combine the features of a mini-computer, and a CRT with touch sensitive input, while still providing a friendly environment to both operator and programmer alike.

The 1720A is unique. It is a full computer. It is rackmountable. It has the best performance/price ratio for controller applications. The 1720A is different from a desk top computer/calculator because the 1720A has been designed as a controller, to work in a controller environment.

The 1720A offers Rackmounting, Detachable Keyboard, Battery Backup, CRT Attributes, Touch Sensitive Display, Character Graphics, a built-in Floppy Disk, additional IEEE-488 Bus Commands, Virtual Arrays, Electronic Disk, additional RS232 and IEEE Ports, Integer Variables, and two character variable names, which are not found on the 9825A. It is for these reasons that FLUKE discourages their customers from doing exact translations of HPL programs into 1720A FLUKE BASIC programs. Surely everyone will want to take advantage of as many of these additional features as possible for their own applications. Of course this manual could be used to write one-to-one HPL to FLUKE BASIC translations, but it is not so intended. This manual is intended as a learning tool for those programmers who are familar with HPL and want a fast way to learn 1720A FLUKE BASIC.

The best way to become knowledgeable enough to add enhancements for the 1720A to an existing HPL program, is to re-write that program from the beginning. In this way you will be best qualified to decide where to place these enhancements in your program.

Of course you could write a program to translate HPL into FLUKE 1720A BASIC, but you would spend more time disecting a translated program to find out where to add the enhancements than you would if you re-wrote the program by hand to begin with. What better way to re-familiarize yourself with a program, so you can know where to add enhancements, than to re-write it and add the enhancements as you see fit.

1.2 HOW TO USE THIS MANUAL

This manual is written for the programmer who has experience in programming Hewlett-Packard's 9825A Desktop Computer. It is intended to be used as a shortcut in learning how to program the Fluke 1720A Controller. It does this by taking advantage of the programmer's knowledge of HPL.

It is assumed that the programmer who will use this manual is thoroughly familiar with the operation and programming of the 9825A and its associated peripherals. It is not the intent of this manual to instruct the programmer in the use of the HPL language. All HPL mnemonics are listed in alphabetical order in each ROM section along with the full word equivalent of the mnemonic. The programmer should refer to the appropriate 9825A programming manual for clarification of any unfamiliar HPL statement or syntax. A partial list of these manuals appears below.

Manual	HP Part Number
9825A Operating and Programming	09825-90000
String Variable Programming	09825-90020
Advanced Programming	09825-90021
General I/O Programming	09825-90024
Extended I/O Programming	09825-90025
9885M Flexible Disk Programming	09825-90000
Systems Programming	09825-90027

Each section lists all of the HPL statements and/or functions for a specific ROM along with its equivalent in 1720A BASIC. Exact 1720A BASIC equivalents are presented immediately next to the HPL statement or function. HPL statements or functions which equate to more than one or two statements in 1720A BASIC are referenced to subsequent pages in that same section where the complete equivalent is given.

This manual can speed up the learning process by directing the programmer to the exact mnemonic equivalent he is looking for. It is not all that obvious, for example, that the HPL mnemonic <beep> is exactly equivalent with the 1720A BASIC statement <PRINT CHR$(7)>.

In most cases the application of a 1720A BASIC statement will be obvious. The Fluke 1720A Programming Manual (Fluke Model No. S1703: Fluke Part No. 518670 should be referred to when further explanation is needed.

In appropriate places, an example of an HPL program and its equivalent 1720A BASIC program is presented at the end of the ROM section to illustrate the use of the statements/functions covered in that section. Descriptive comments follow exclamation (!) marks in the example 1720A BASIC Code. <> is used to enclose an element of code. The <> is not part of the code. The programs listed in the table of programs are available on a pre-recorded disk.

The programmer should read the following sections in this manual first:

2 General Description of the Fluke 1720A
2.2 Power Up Procedure
2.3 Access the Basic Interpreter Mode
3.1 Equivalent Programming Capabilities Features
3.2 Essential Programming Differences

Sections 3.1 and 3.2 will show the programmer where previous experience will provide immediate usable knowledge and where the programmer must acquire new knowledge. Sections 2, 2.2, and 2.3 indicate what has to be done to the 1720A to enable the Basic Mode and show the programmer how to enter, edit and execute the Basic statements and functions he finds in this manual. After these sections have been read, the remainder of this manual may be read in any order as the need arises.

The OLD and SAVE commands as well as the DIM, INPUT, INPUT LINE, OPEN, PRINT and PRINT USING statements are constructed to include device numbers and channel numbers. Default values are assumed which reference system devices when these numbers are not specified. For a more complete discussion of this subject, refer to Page 5B in the appendix of this manual. This discussion also applies to the File Utility Program (FUP).

Section 4 titled Programming Tips points out pitfalls that programmers who are used to using HPL may fall into. This section also points out some useful shortcuts and enhancements which are not available with 9825A HPL, but are available on the Fluke 1720A.

The appendix provides additonal information which may prove useful to the programmer in understanding the programs in this manual.

The following 9825A ROMs have no equivalent statements or functions on the 1720A:

Matrix ROM
Plotter ROM
9872A Plotter ROM

The Command File, Console Monitor and File Utility Program (FUP) have their similar functions represented in the 9825A ROM operating system. It is not necessary to understand these 1720A operating modes in order to write 1720A BASIC, although the programmer will eventually want to gain a thorough understanding of these modes in order to take advantage of the full power of the 1720A. These modes are described in section 2.4 of this manual.

SECTION 2 GENERAL DESCRIPTION OF THE FLUKE 1720A CONTROLLER

The Fluke Model 1720A Instrument Controller performs the control and computation functions in instrumentation systems. The 1720A is designed to address two different environments of employment. One is the application program development environment (the "Programmer Environment"), the other the application execution environment (the "Operator Environment").

The key features include:

Two IEE-488 standard 1978 digital interfaces for programmable instrumentation or peripheral devices.

BASIC language enhanced for instrument control.

Alphanumeric CRT display of 16 lines by 80 characters or 8 lines 40 double size characters, selectable under program control.

Soft keyboard capability implemented with a touch-sense, optically transparent keyboard overlaying the CRT display. Six rows x 10 columns of touch-sense keys (60 keys).

Soft loaded software for performance flexibility.

One double density single-head, mini-floppy disk drive with a data capacity of 1750K bytes (K = 1024). 348 blocks are accessible to the user where 1 block is .5K (512) bytes.

Electronic Disk providing a data capacity of 128K bytes extendable to 256K bytes. 254 blocks (extendable to 510 blocks) are accessible to the user where 1 block is .5K (512) bytes.

Detachable alphanumeric keyboard. Two-Key Rollover. All multiple key lockout. Automatic repeat when key is held down.

Two RS-232-C Interfaces, Type D, for full duplex asynchronous communication.

*60K bytes of accessible read-write random access byte parity memory; (K =1024, 1 byte = 8 bits, 1 word = 2 bytes).

Utility programs for ease of operation.

*Approximately 24,000 bytes (12,000 words) available for user's BASIC program and data.

2.1 1720A ENHANCEMENTS (FEATURES NOT AVAILABLE ON 9825A) (ALSO REFER TO PROGRAMMING ENHANCEMENTS IN SECTION 4.2)

Table 2-1

The following table shows 1720A features which are not available on the 9825A and compares these features with the nearest comparable feature on the 9825A:

	1720A	9825A COMPARISON
1.	CRT Display a. 80 characters x 16 lines b. Double size characters c. Reverse video d. Blinking characters e. Underlined characters f. Page mode g. Character Graphics	1. 32 Character LED Display a. Blinking characters
2.	User defined Touch-Sensitive Display - 60 key, keyboard on face of CRT (Operator's keyboard) CRT provides dynamic overlay	2. 24 special function keys on programmer's keyboard; requires overlay and use of shift key
3.	Programmer's keyboard is detachable	3. Keyboard is fixed
4.	Operating and programming functions are distinctly separate from each other	4. Operating and programming functions are together
5.	Operating system is soft loaded. This provides an inexpensive way to support the operating system	5. Operating system is on ROM. Expensive to replace and/or update
6.	Virtual arrays (see note below)	6. No virtual arrays
7.	Strings, advanced programming, General I/O, and flexible disk statements are standard	7. Must purchase four additional ROMS

NOTE: Virtual arrays differ from ordinary arrays in that they reside on the storage medium (floppy disk or Electronic Disk) rather than in the 1720A read/write memory. Virtual arrays are similar, however, to ordinary arrays because they can be operated on directly without using PRINT and INPUT statements. When the associated file is closed, virtual arrays are automatically stored on the disk. See programs VARAY1 and VARAY2 in section 3.15

8. 24,000 bytes of BASIC R/W memory available to programmer NOTE: This may change slightly as system is updated.	8. 6,844 bytes standard; 31,420 bytes maximum (less when ROMS are installed).
9. Floppy disk is standard	9. Floppy disk is optional
10. Optional Electric Disk for high speed mass storage - see item 15 in section 4.2	10. Electronic Disk not available
11. Battery back up for mass storage (in case of power failure)	11. No battery backup for mass storage
12. Two RS-232 ports are standard - see item 14 in section 4.2	12. RS-232 ports optional with interface cards
13. Two IEEE ports are standard- see item 13 in section 4.2	13. HPIB ports are optional with interface cards
14. Real time clock	14. Real time clock optional
15. Calendar (day-mo.-year)	15. Calendar optional
16. 15 printable significant digits	16. 12 printable significant digits
17. Commas can easily be placed in numbers via PRINT USING statement	17. No easy way to place commas in numbers
18. Line numbers can be assigned by programmer according to some meaningful arrangement (use REN command)	18. Lines automatically numbered consecutively; cannot be renumbered by programmer
19. Remarks can appear at end of each program line	19. Labels used for remarks
20. System variables ERR and ERL for error number and line number	20. System variables not standard
21. Indexed GO TO	21. Must use jump statement
22. Integer Variables	22. No Integer Variables
23. Two character variables (letters and numbers)	23. Single letter variables
24. Tracing of variables	24. Variables cannot be traced except by line No.

25. CONT TO command	25. No equivalent
26. String arrays have no maximum character length	26. Once the maximum character length has been dimensioned it cannot be changed
27. STEP command executes a statement at a time	27. STEP key executes a line at a time, regardless of how many statements are in that line
28. Desktop or rack mountable	28. Desktop only
29. Command File - the command file is automatically loaded at power up or when restart is pressed. It can be programmed to implement any of the system software as well as basic and can be programmed to call for a series of different system and basic programs	29. Similar to the automatic loading of file 0 when the calculator is powered up with a tape cartridge inserted

2.2 HOW TO POWER UP

POWER UP PROCEDURE

1. Make sure power cord is plugged in.
2. Turn on power switch (located on rear of the 1720A).
3. Read the directions on the flexible disk envelop and in the 1720A User's Manual for care and use of the flexible disk.
4. Gently insert disk label side up, slot side first (see illustration below).

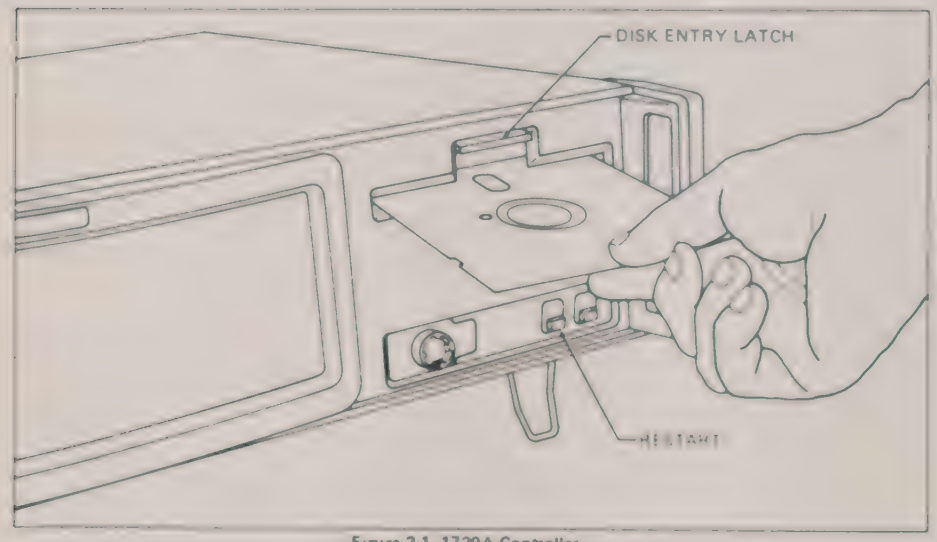


Figure 2-1. 1720A Controller

5. Gently close disk entry latch (see figure 2-1 above); using several partial closures to seat the disk.
6. Press RESTART (see above illustration).
7. "Console Monitor" is displayed.
8. # is displayed.
9. Type B.
10. Press RETURN.
11. "BASIC" is displayed.
12. Wait for "ready" to be displayed.
13. When "Ready" is displayed the 1720A is in the BASIC Immediate Mode.

2.3 ACCESS THE BASIC INTERPRETER MODE

The 1720A is in the Immediate Basic Mode after Basic is called for and accessed. Keyboard arithmetic can be done in the immediate mode by using the PRINT statement and assignments can be done directly. For example:

```
Type:     A=2 \ B=8 ! assignment
Press:    RETURN
Type:     PRINT A^B   !2^8
Press:    RETURN
Display   shows:  256
```

Basic programs may be entered in the immediate mode or the EDIT mode. Cursor editing on previously stored lines can only be done in the EDIT mode. To get into the EDIT mode:

```
Type:     EDIT
Press:    RETURN
```

Line numbers must be typed in, i.e., they are not automatically entered by the system. Lines are stored by directing the cursor to the end of the line and pressing RETURN.

To get from EDIT to the immediate mode:

```
Simultaneously press:          CTRL C
```

To run a program:

```
Type:  RUN
Press: RETURN
```

2.4 SYSTEM SOFTWARE MODE CONFIGURATION

Program Start-Up

Control is passed to one of two programs after configuration and turn-on of the unit. These programs are the Console Monitor and the Command File Processor.

Command File Processor

The Command File Processor (CFP) is a system module that allows the user to automate certain programming procedures. Its basic function is to accept character strings which are normally obtained from the console terminal. The CFP gets these strings from a special file. Every time a program expects a line of input, the next line from the command file is extracted. The CFP accesses only one file: COMMND.SYS. If this file is not present on the system, the CFP defaults to the console terminal for its input stream.

Execution

The CFP is automatically invoked when the system is powered up or when the RESTART button is pressed.

2.4.1 Command File Generation

Since the command file consists of more or less random character strings, it cannot be edited using the BASIC editor, since that editor expects strings which comply with the BASIC syntax. Therefore, a small BASIC program is provided which enables the user to generate a command file. This program consists of two parts. The first part is a collection of DATA statements which represent the content of the command file. The second part reads those DATA statements and generates the command file from them. If a change needs to be made to the command file, the user simply edits the data statement portion of this program then runs the program and a new command file is obtained.

Power Up Sequence

When initially supplied to the user, two standard command files are on the system disk. One command file is used when the system is loaded from the floppy disk. This command file will call the TIME program to request time and day information from the operator. Then the Electronic Disk will be formatted and certain system files will be copied onto the Electronic Disk. Then it will make the electronic disk the system device. Finally, the BASIC Interpreter will be invoked. The second command file will be used when the system is loaded from the Electronic Disk. The only action taken by this command file is to invoke the BASIC interpreter.

The Command File can accomplish those operations that are normally performed from the keyboard of the 9825A:

ldp, list #, drive, repk, init, copy

The Command file can execute the Console Monitor tasks, as well as any of the File Utility Program Tasks.

Selection of Basic Programs can be controlled by the Command File with control return to the Command File via the Basic Statement "EXIT".

The Command File takes charge as soon as the operator presses RESTART on the 1720A. If not Electronic Disk is present, a disk with an operating system must be inserted prior to pressing RESTART. See item 4 in section 4.3 for Command File Programming.

2.4.2 Console Monitor - CONMON

This non-resident portion of the system software is loaded from the system device into main RAM and started following:

a. Completion of the execution of self-check and startup code following "power-on" or RESTART and the Command File Processor.

b. Activation of P on the console terminal.

c. or, explicit exit from a utility program or language processor.

The CONSOLE MONITOR introduces itself on the console terminal with the following message and user prompt:

```
CONSOLE MONITOR VERSION x.y*
#
```

In response to the prompt character, the user enters the name of the task he wishes to have executed followed by <RETURN>. The allowable tasks and the names used to invoke them are listed below. The portion of the name contained in brackets ([]) is optional. Simultaneously pressing CTRL P may be typed at any time to return to prompt (#) for new command.

TABLE 2-2

CONSOLE MONITOR TASKS

Task Name	Task Description
? <RETURN>	Help to get a brief version of the following list.
S[ET] <RETURN>	Select a program to set the baud rates of the RS-232-C ports.
B[ASIC] <RETURN>	Select the BASIC interpreter.
F[UP] <RETURN>	Select the File Utility Program.
T[IME] <RETURN>	Reset system hardware clock.
<filename><RETURN>	The name of a memory image file (output of SAVE or LINK) to be loaded and executed.

2.4.3 File Utility Program (FUP)

The File Utility Program is a file transfer and file management program for the user. The user can utilize this program to transfer files between any of the I/O devices and to merge, rename, delete files as well as to restructure diskettes.

PROGRAM 1

FUP.HLP

Summary of Options in FUP with examples are as follows:

/A	Assign System Device	EDO:/A or MFO:/A
/B	Binary File Transfer	MFO:FUP.CIL = MMO:FUP.CIL/B
/D	Delete a File	TEST.TAP/D or EDO:TEST.1,TEST.2,TEST.3/D
/E	Extended Directory Listing	MFO:/E or EDO:/E or /E
/F	Format a Disk or the E-Disk	MFO:/F or EDO:/F
/L	List a Directory	MFO:/L or KB1: = EDO:/L
/M	Merge ASCII Files	TEST.NEW = TEST.1,TEST.2,TEST.3/M
/P	Pack a Device	MFO:/P or EDO:/P or /P
/R	Rename a File	TEST.1 = TEST.OLD/R
/S	Search for Bad Blocks	MFO:/S or /S
/T	Turn off error checking	EDO:TEST.BAD = MFO:TEST.BAS/T
/W	Whole Copy or Partial Copy	EDO: = MFO:/W or MFO: = MMO:BASIC.CIL/W
/X	Exit from FUP	/X
/Z	Zero a Directory	EDO:/Z or MFO:/Z or /Z

TO COPY DISK

CTRL P TO GOTO MONITOR MODE

TYPE "FUP" TO ENTER FILE UTILITIES

INSERT NEW DISK IN DRIVE

TYPE MFØ:/F TO FORMAT DISK

INSERT SOURCE DISK MFØ:/W TO W: CONTENT

TYPE MMØ:=MFØ:/W TO COPY DISK TO MEMORY

INSERT DESTINATION DISK

TYPE MFØ:=MMØ/W TO COPY MEMORY TO DISK

IF DISK DOES FIT INTO MEMORY

CLEAR MEMORY BY TYPING MMØ:/Z

INSERT SOURCE DISK

TYPE MMØ:=MFØ:{FILENAME}/W

{FILENAME}=LAST FILE COPIED FROM PREVIOUS CASE

ENTER DEST DISK

TYPE MFØ:=MMØ

NOTES (FOR FIGURE 2-2 ON NEXT PAGE):

1. To startup system, a) insert floppy disk, b) turn on power

2. System Self-Test checks presence and workability of all system modules. Programmed on ROM. Also includes facility to load the operating system.

3. Error Messages - when error condition is corrected, the Self-Test is restarted.

4. FDOS is the Fluke Disk Operating System. The Command File is a file named COMMND.SYS which is programmed by the user to initialize the system, run BASIC programs or whatever else is necessary for system operation.

5. If a Command File is present, a flag is set. The flag is now checked to determine if input will be from a Command File or from the keyboard.

6. The Console Monitor (CONMON) is the 'director' for calling in System Tasks.

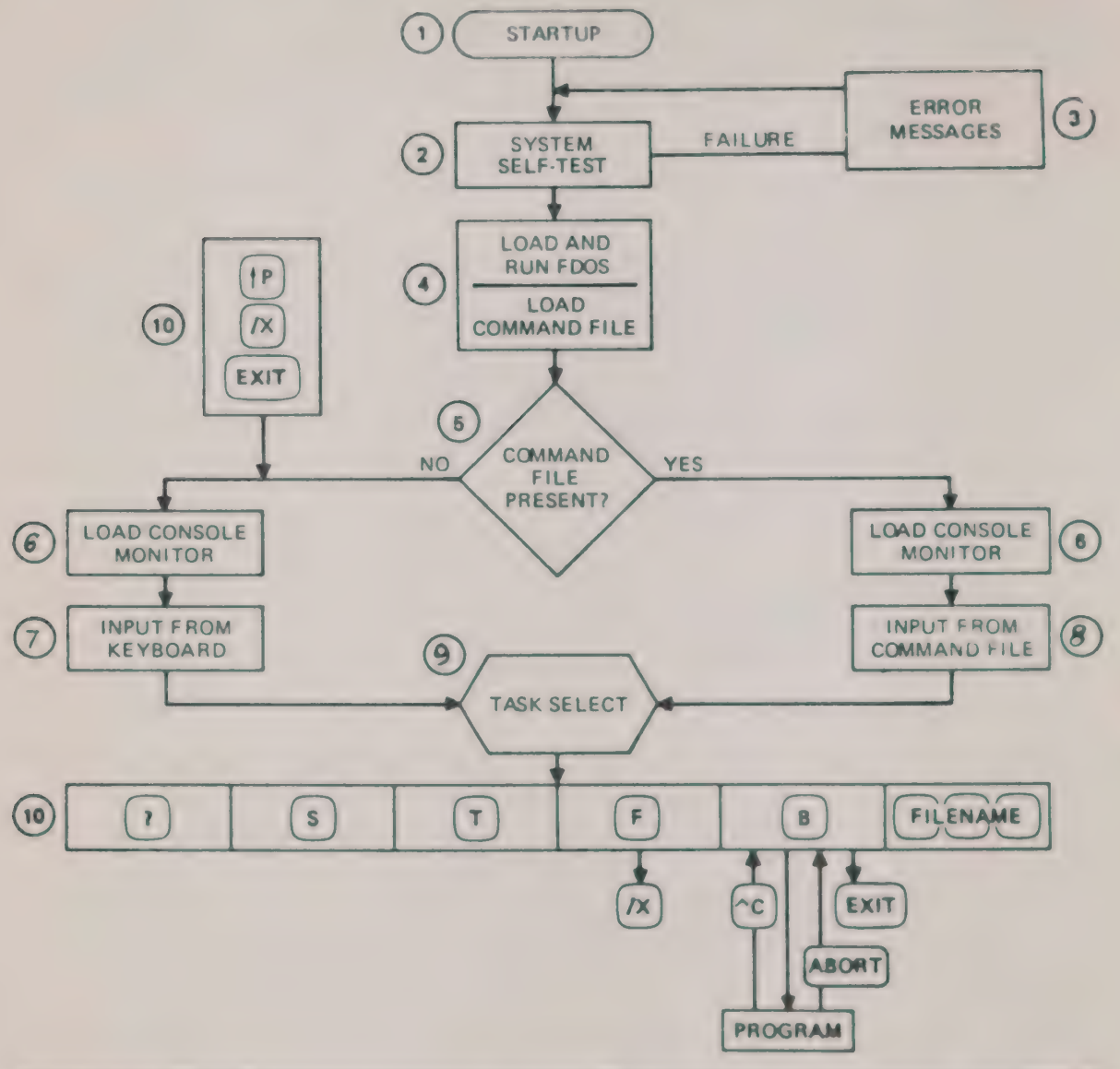
7-8. Input to the System may be from either the keyboard(7) or from a Command File (8).

9. A task is selected by either Command File or keyboard. See Task Table.

10. Exits:
    ^P returns control to the Console Monitor
    ^C returns control from a BASIC program to the Basic Interpreter
    /X returns control to the Console Monitor from the File Utility Program
    EXIT returns control to the Console Monitor from the BASIC Interpreter
    ABORT returns control from a BASIC program to the BASIC Interpreter

SYSTEM SOFTWARE CONFIGURATION FLOW CHART

FIGURE 2-2



TASK	DESCRIPTION	EXITS
?	Displays a 'menu' of the various tasks performable.	None Required
S	Set Baudrate Utility	None Required
T	Set Time/Date Utility	None Required
F	File Utility Program	CTRL P, /X
B	BASIC Interpreter	CTRL P, EXIT
FILENAME	Uses any of the Assembly Language Programs.	As Programmed

1720A Figure 2-2 Software Block Diagram (See notes on Previous Page)

3.1 EQUIVALENT PROGRAMMING CAPABILITIES (1720A To 9825A)

The 1720A has the following features which are equivalent to the 9825A:

1. 80 character programming line
2. Programming lines may contain multiple statements
3. Same mathematical hierachy
4. Full ASCII character set
5. Live keyboard can be disabled
6. Numbered error messages
7. Simple variables
8. Array variables
9. String variables
10. Absolute branching
11. Interfacing and controlling capability
12. Math functions
13. Language equivalents
14. Calculated GOSUB and calculated GOTO
15. Program editing
16. Mass storage
17. Automatic line renumbering of GO TO, GOSUB, RESTORE, TRACEON and STOP ON addresses during line renumbering
18. Linking of programs
19. Operators
20. Random number
21. Program tracing
22. Display of program lines, prompts and results
23. String operations

3.2 ESSENTIAL PROGRAMMING DIFFERENCES (9825A COMPARED TO 1720A)

		9825A	1720A
1.	Language	HPL	BASIC
2.	Operating System	ROM	Soft loaded
3.	Storage	Tape or optional disk	Disk or optional Electronic Disk
4.	Keyboard	Fixed	Detachable
5.	Sub Program Variables	Local	Global
6.	Common Statement	Not Needed	Needed
7.	Labels	Allowed	Not Allowed
8.	Line Numbering	Automatic & in steps of 1	Programmer's choice (no automatic numbering)
9.	Colon	After Line Number	Not allowed
10.	Statement Separator	;	\
11.	Array Dimensions	Programmer's Choice; 1st element number is 1, unless otherwise specified	All arrays have 1st element start at 0
12.	Maximum Numbers of Array Dimensions	Limited by available memory only	2 Dimensions*
13.	Assignment	A--->B--->C	B=A\C=A
14.	End of Line	Requires ┤(Automatic)	No visable End-Of-Line Character
15.	String concatenation	&	+
16.	π	π	PI
17.	MNEMONICS	lower case	UPER CASE

*A single dimensional array can be programmed for N dimensions (see program 2 in section 3.7 and Appendix 5A.)

	9825	1720A
18. Relative Branching	Allowed	Not Allowed
19. Implied Multiplication	Allowed	Not Allowed
20. Truth Statements	(A=B)= 1 for true	(A=B)= -1 for true
21. Statement Storing	STORE key only	Implied scrolling
22. Undimensioned Variables	Optional in dim statement	Not allowed in dim statement
23. gto (GO TO)	Last statement executed	Executed when encountered
24. gsb (GOSUB)	Last statement executed	Executed when encountered
25. String Length	Must be dimensioned	Has no limit except available memory
26. Keyboard Arithmetic	Type in function and press EXECUTE	Type PRINT, type in function, press RETURN
27. String Array: Maximum Character Length	Must be dimensioned	Dynamic
28. Array Brackets	Square	Round
29. Imbedded Assignment	Allowed	Not Allowed

USER STORAGE

9825A CASSETTE	9885 DISK	1720A DISK
280,000 Bytes max (actual capacity depends on file length)	400,000 Bytes 2010 Records 200 Bytes/Record 67 Tracks	179,200 Bytes 350 Blocks 512 Bytes/Block 40 Tracks

3.3 PROGRAMMING ENHANCEMENTS

The following is a list of 1720A programming and editing capabilities for which there is nothing similar in HPL. The programmer needs to consult the 1720A Programming Manual (Fluke Model No. S1703: Fluke Part No. 518670) to become familiar with these enhancements in order to take full advantage of the power of the Fluke 1720A.

COMMANDS

CONT TO — Program is halted at a specific line number

REN — used to renumber lines

Examples:
```
REN 540-697 AS 550 STEP 2
REN 9041 AS 32767
```

SAVEL — saves the lexically analyzed form of the BASIC program currently in memory. Allows faster program loading.

Examples:
```
SAVEL "TEST3"
SAVEL "EMO:PROG9"
```

device / program name

STATEMENTS

COM — the common statement. Allows variables to be passed from one program to another.

Example:
```
<line no.> COM A,B (10),C (2,8)
```

DATA — Supplies a list of data values to be accessed by READ statements in the BASIC program.

Example:
```
<line no.> DATA 4.3,
"STRING", 85,49,75,04,10,NEWSTRING
```

EXIT — Halts execution of BASIC interpreter and returns control to the console monitor

Example:
```
<line no.> EXIT
```

IF-THEN-ELSE — allows another option in an IF statement

Example:
```
<line no.> IF B=A THEN PRINT "EQUAL"
ELSE PRINT "NOT EQUAL"
```

ON KEY GO TO & OFF KEY	<line no.> Specifies that if the operator's keyboard is activated, the program control should be transferred to the specified line no. OFF KEY will disable the ON KEY interrupt. These statements are used with the CRT touch-sense keys.
RANDOMIZE	Uses System clock to initialize random number generator
READ	Assigns data for DATA statement to the list of variables Example: <line no.> READ A,B$, C%, F1, R2, B(25), D(0..43)
RESTORE	Resets pointer to program DATA statements to DATA statement in program Example: <line no.> RESTORE [<line no.>]
RESUME	Returns control from interrupt or error handling routine. A RESUME without a line number will return control to the statment which would have been executed had the interrupt or error not occurred Example: <line no.> RESUME [<line no.>]
CRT DISPLAY ATTRIBUTES	See Section 4.2
KEYBOARD CONTORL CHARACTERS	See Programming Manual
ESCAPE and ESCAPE SEQUENCES	See Section 4.2
CURSOR CONTROL COMMANDS	See Section 4.2
VIRTUAL ARRAYS	See Programs 18 and 19 in Section 3.17

3.4 RULES FOR TRANSLATING 9825A HPL INTO FLUKE 1720A BASIC

1. Change all relative addresses and label addresses to absolute addresses on the HPL listing.

2. Make a list of all variables to be used in a COM (common) statement or virtual array if this program is to be linked to other programs.

3. Assign new line addresses by increasing each old address by a factor of 10 (exception: line No. 0), e.g., 0=5, 1=10, 2=20, 3=30, etc. Later on a meaningful address numbering system can easily be assigned using the REN (renumber) command.

4. Note the associated HPL line number in remarks adjacent to the line of BASIC.

5. Change all square brackets to round brackets.

6. Reverse the order of all variable assignments and replace the "--->" with "=", e.g., A--->B becomes B=A and A--->B--->C becomes B=A\C=A

7. Get rid of all imbedded assignments by breaking them up, e.g., if (J+1--->J)>M;gto20 becomes J=J+1 \ IF J>M GO TO <line No.>

8. Do not place a colon after a line number; you may leave a space before typing the first statement.

9. Replace ";" statement separators with "\".

10. Use the multiplication operator in place of implied multiplication, e.g., AB becomes A*B.

11. Make all truth statements negative, e.g., (A=B)(A+B+C)--->V becomes V=-(A=B)*(A+B+C).

12. Remove any undimensioned variables from "dim" statements and allocate them by setting them equal to 0, e.g. dim A,B,C becomes A=0\B=0\C=0.

13. Refer to statement and function equivalents given in this manual.

14. Verify proper order for multiple relational operators, e.g., A=<B becomes A<=B.

15. Use the same array dimensions and ignore subscript 0 if available memory will allow this; otherwise subscripts must be re-assigned to include 0.

16. If degrees are used be sure to convert to radians (and vice versa) as shown in this manual). Degrees=Radians*180/π

17. Replace all "fxd", "flt" and "fmt" statement with the PRINT USING statement.

18. Replace all imbedded "jmp" statements with FOR and NEXT loops.

19. Convert any cassette operation statements to disk statements as shown in this manual. Make sure you read the information on the OPEN and CLOSE statements (see section 3.13 and Appendix 5B in this manual) to handle data storage.

20. All "IF" statements must include a "THEN" or a "GO TO" which takes the place of the HPL semicolon in the "IF" statement.

3.5 SPECIAL KEY EQUIVALENTS

TABLE 3-I

The following table equates the 9825A Keyboard with the 1720 Keyboard:

3.5.1 Control Keys

9825A	1720A BASIC	Mode
CLEAR	Simultaneously Press: CTRL + T	Immediate
CONTINUE	Type: CONT Press: RETURN	Immediate
EXECUTE	Press: RETURN	Immediate
RUN	Type: RUN Press: RETURN	Immediate
STOP	Simultaneously Press: CTRL + C	Immediate
STORE	Press: RETURN or ^* or v*	Editor
SHIFT	Press: SHIFT	All
SHIFT LOCK	Press: CAPS LOCK	All

*Scrolling the Display. A line is automatically stored when the cursor is directed to another line.

3.5.2 Special Function Keys

9825A	1720A BASIC	Mode
f_0 f_6 f_1 f_7 f_2 f_8 f_3 f_9 f_4 f_{10} f_5 f_{11}	Replaced by 60 "TOUCH SENSE" Keys on CRT	Programmable

3.5.3 System Command Keys

9825A	1720A BASIC	Mode
RESET	Press: ABORT	All
PRT ALL	All messages normally appear on CRT and are saved until paged out. Alternately, messages can be sent to a printer	All
STEP	This command legal only after STOP ON or CONT TO command. Type: STEP for each statement, Press: RETURN	Immediate
LIST	List entire program Type: LIST Press: RETURN or List portion – Type: LIST <line No.>–<line No.> Press: RETURN	Immediate
REWIND	Not needed with disk	NA
LOAD	Type OLD '<file name>' Press RETURN	Immediate
RECORD	Type: SAVE '<up to 6 character file name>' Press: RETURN	Immediate
ERASE	Type: DELETE ALL Press: RETURN	Immediate
a	Press: RESTART	All
v	Not Available	NA
k	Not needed	NA
fn	Not needed	NA

3.5.4 Line Editing Keys

9825A	1720A BASIC	Mode
DELETE	Type: DELETE < line No. > – < line No. > or *Cursored line - Press: DEL LINE	Immediate / Editor
INSERT	To create space to insert a line: 1. Move cursor to far right of line which will precede 2. Press: RETURN	Editor
RECALL	Not Available	NA
FETCH	Type: EDIT < line No. > Press: RETURN	Immediate

Every character to the right of the cursor (including the cursored character) gets deleted

3.5.5 Character Editing Keys

9825A	1720A BASIC	Mode
BACK	Press: <	Editor
FWD	Press: >	Editor
DELETE	Press: DEL CHAR or DELETE deletes adjacent left characters	Editor / Editor & Immediate
INS/RPL	*Replace: Not directly available Insert: Takes place automatically	

Replace is a two step operation.

1. *The unwanted character must be deleted.*
2. *The new character must be inserted.*

} *Either order*

3.5.6 Display Keys

9825A	1720A BASIC	Mode
↓ ↑	Press V ^	Editor
← →	Not necessary: Entire line is displayed	Editor

Statement Separator

9825A	1720A
;	\

3.5.7 Miscellaneous Important Keys

9825A	1720A
:	Line numbers are not followed by a colon, however, a space is recommended
↑	^ for exponentiation
π	Type PI
ENTER EXP	Type E
RESULT	Not available
→ 2→A	A = 2 Note: → becomes = "2" and "A" reverse
=	(A = B) truth Statements are allowed *

*(A = B) = 0 for false (A = B) = −1 for true

3.6 OPERATOR EQUIVALENTS

TABLE 3-2
OPERATOR EQUIVALENTS

The following table equates the HPL operators with the BASIC operators:

Assignments:

	9825A HPL	1720A BASIC
	5 ---> A ---> B	A=5 \ B=5

Arithmetic

Operation	9825A HPL	1720A BASIC
Add	A + B	A + B
Subtract	A - B	A - B
Multiply	A * B	A * B
Implied Multiply	AB	A * B
Divide	A/3	A/3
Exponentiate	X↑Y	X^Y
Modulus	Amod 256	Programmed equivalent is given below:

```
<line number> DEF FNM (X)=X-256*INT(X/256)!   defined function see
<line number> B=FNM (A) !  B=Amod256          section 3.12
```

Relational

Operation	9825A HPL	1720A BASIC
Equal	X = Y	X = Y
Greater than	X > Y	X > Y
Less than	X < Y	X < Y

Greater than or Equal	X>=Y or X=<Y	X>=Y order dependant
Less than or Equal	X<=Y or X=<Y	X<=Y order dependant
Not Equal	X#Y or X><Y orX<>Y	X<>Y order dependant

Logical

	9825A HPL	1720A BASIC
	and	AND
	or	OR
	xor	XOR
	not	*NOT

*CAUTION: be sure to read item 22 under PROGRAMMING PITFALLS

3.7 USER ASSIGNED VARIABLE EQUIVALENTS

Table 3-3

The following table equates HPL variables with BASIC variables:

9825 HPL	1720A BASIC
Simple Variables:	
A through Z	A through Z also A (any letter) though Z (any letter), e.g., AA, AB, BA, BZ and A(0 thru 9) through Z (0 through 9),e.g.,A1,A2,A9
Integer Variables:	
None	follow any of the above with %
Array Variables: A through Z	Same names as simple and integer variables
[dimensions] use square brackets. Only limit to number of dimensions is available memory	(dimensions) use round brackets.Limit: 2 dimensions-see program 2 for algorithm to expand number of dimensions to 3; see appendix 5A for algorithm to expand to N dimensions
String Variables:	
Follow any of the above variables with $; limit of 2 dimensions	Follow any of the above variable names with $
r Variables r0 through rXXX	Suggest the use of a dimensioned variable, R(N%) DIM R(N%) yields R (0%) through R(N%)

Illegal Variable Names

IF, ON, OR, TO, FN, LN, PI, AS

Note: All 1720A dimensions start at 0

Example:

TABLE 3-3a

DIMENSION COMPARISON

	HPL		1720A BASIC	
dim	A [5,2]	yields 10 elements	DIM A (5,2)	yields 18 elements
	A [1,1]		A (0,0)	
	A [1,2]		A (0,1)	
	A [2,1]		A (0,2)	
	A [2,2]		A (1,0)	
	A [3,1]		A (1,1)	
	A [3,2]		A (1,2)	
	A [4,1]		A (2,0)	
	A [4,2]		A (2,1)	
	A [5,1]		A (2,2)	
	A [5,2]		A (3,0)	
			A (3,1)	
			A (3,2)	
			A (4,0)	
			A (4,1)	
			A (4,2)	
			A (5,0)	
			A (5,1)	
			A (5,2)	

NOTE: DIM (dim) statements are explained in section 3.11

```
1 ! '3DARAY',            3 DIMENSIONAL ARRAY
7 ! VERSION 1.0, 19 NOVEMBER 1979
10 !
20 !*************************************************************************
30 !
100  ! THIS PROGRAM DEMONSTRATES ONE WAY OF CREATING A THREE DIMENSIONAL
110  ! ARRAY FROM A SINGLE DIMENSIONAL ARRAY
1120 !
2000 PRINT"Max. Dimensions, A,B,C";        ! PROMPT
2010 INPUT AZ,BZ,CZ                        ! INPUT MAX. DIMENSIONS
2020 DIM V(AZ*BZ*CZ-1Z)                    ! DIMENSION EQUIVALENT ARRAY
2030 FOR IZ=1 TO AZ                        ! FIRST SUBSCRIPT
2040 FOR JZ=1 TO BZ                        ! 2ND SUBSCRIPT
2050 FOR KZ=1 TO CZ                        ! THIRD SUBSCRIPT
2060 GOSUB 2130               ! V(IZ,JZ,KZ)=100Z*IZ+10Z*JZ+KZ  LOADS ARRAY
2070                          ! ELEMENTS WITH THEIR OWN SUBSCRIPTS
2080 PRINT V(DZ),IZ,JZ,KZ,DZ               ! MATCH CONTENTS WITH SUBSCRIPTS
2090 NEXT KZ
2100 NEXT JZ
2110 NEXT IZ
2120 GOTO 2180                             ! END
2122 !
2130 !          SUBROUTINE TO EQUATE D TO I,J,K
2135 !
2140 DZ=((IZ-1Z)*BZ*CZ+(JZ-1Z)*CZ+KZ-1Z)
2150 V(DZ)=100Z*IZ+10Z*JZ+KZ               ! LOAD EQUIVALENT ARRAY ELEMENT
2160                                       ! WITH ITS EQUIVALENT SUBSCRIPTS
2170 RETURN
2180 END
4000 !
4010 !*************************************************************************
4020 !
4030 !
4040 !
4050 !
4060 !
4070 !
4080 !
4090 !
4100 !  THE EXAMPLES ON THE NEXT PAGE SHOW HOW THE SAME PROGRAM WOULD
4110 !  BE PROGRAMED USING HPL
4120 !
4130 END
```

Program 2 (Program for3-D array)

```
V[I,J,K]  I  J  K
   111    1  1  1
   112    1  1  2
   113    1  1  3
   114    1  1  4
   121    1  2  1
   122    1  2  2
   123    1  2  3
   124    1  2  4
   131    1  3  1
   132    1  3  2
   133    1  3  3
   134    1  3  4
   211    2  1  1
   212    2  1  2
   213    2  1  3
   214    2  1  4
   221    2  2  1
   222    2  2  2
   223    2  2  3
   224    2  2  4
   231    2  3  1
   232    2  3  2
   233    2  3  3
   234    2  3  4
```

```
0: ent "Max.
 Dimensions",A,
 B,C
1: dim V[A,B,C];
 prt "V[I,J,K]
 I J K"
2: 0→I
3: I+1→I;0→J
4: J+1→J;0→K
5: K+1→K
6: 100I+10J+K→V[
 I,J,K];fmt 1,
 2x,f4.0,2x,f2.0
 ,f2.0,f2.0
7: wrt 16.1,V[I,
 J,K],I,J,K
8: jmp -4(K<C)+1
9: jmp -6(J<B)+1
10: jmp -8(I<A)+
 1
11: end
*21754
```

FILE 2

```
V[D]   I  J  K   D
 111   1  1  1   1
 112   1  1  2   2
 113   1  1  3   3
 114   1  1  4   4
 121   1  2  1   5
 122   1  2  2   6
 123   1  2  3   7
 124   1  2  4   8
 131   1  3  1   9
 132   1  3  2  10
 133   1  3  3  11
 134   1  3  4  12
 211   2  1  1  13
 212   2  1  2  14
 213   2  1  3  15
 214   2  1  4  16
 221   2  2  1  17
 222   2  2  2  18
 223   2  2  3  19
 224   2  2  4  20
 231   2  3  1  21
 232   2  3  2  22
 233   2  3  3  23
 234   2  3  4  24
```

```
0: "3D-ARRAY":
1: ent "Max.
 Dimensions",A,
 B,C
2: dim V[ABC];
 prt "V[D]  I
 J K  D"
3: for I=1 to A
4: for J=1 to B
5: for K=1 to C
6: D+1→D
7: fmt 1,f4.0,
 2x,f2.0,x,f2.0
 ,f2.0,x,f2.0
8: wrt 16.1,V[D]
 ,I,J,K,D
9: next K
10: next J
11: next I
12: gto 17
13: "SUBROUTINE
 to equate D to
 I,J,K":
14: (I-1)BC+(J-
 1)C+K→D
15: 100I+10J+
 K→V[D]
16: ret
17: end
*32713
```

FILE 3

3.8 FLAG EQUIVALENTS

The 9825A has 16 dedicated flages (0 through 15). The HP mnemonic for flag is flg. flg is followed by the flag number, e.g., flg 3. 1720A Basic has no dedicated flags, however, flags can be easily created by use of the DIM statement. By using the DIM statement, memory for flags can be limited to the exact amount of flags required for a particular application.

The 16 HPL flags may be re-created in 1720A Basic by the following statement:

```
<line number> DIM FG%(15%)
```

where DIM = dimension
FG = flg = flag
% = integer (requires less memory than floating point)
15 = allocation of 0 through 15, i.e., dimension

sfg is the HPL mnemonic for set flag. A set flag has the value of 1. For example:

<sfg 3> sets flag 3; consequently flg 3 = 1

Flag 3, or any other flag, may be set using 1720A Basic by the following statement:

```
<line number>FG%(3%)=-1%  ! -1 = True for FLUKE BASIC
                          read item 5 and item 22 in
                          section 4.1
```

cfg is the HPL mnemonic for clear flag. A cleared flag has the value of 0. For example:

<cfg 3> clears flag 3; consequently flg 3 = 0

Flag 3, or any other flag, may be cleared using 1720A Basic by the following statement:

```
<line number> FG%(3%)=0%
```

cmf is the HPL mnemonic for "compliment flag". If flg 3 = 0, it is set to 1, if flg 3 = 1, it is set to 0.

Flag 3, or any other flag, may be complimented in 1720A BASIC in the following way:

```
<line number> FG%(3%) = (FG%(3%)=0%) ! where FG%(3%)=-1
Or 0
```

Note: IF FG%(3) THEN...is a valid alternative to IF FG%(3) = -1 THEN...

To set all flags:

```
<line number> FOR I% = 0% to 15%
<line number> FG%(I%)=-1%
<line number> NEXT I%
```

To clear all flags:

```
<line number> FOR I%-=0% to 15%
<line number> FG%(I%)=0%
<line number> NEXT I%
```

3.9 Special Flags

flg 12 - See tfr statement in section 3.18

flg 13 - Automatically set if no data is entered at an ent or enp statement and the CONTINUE or STOP key is pressed

The following statements will provide the same features in 1720A BASIC:

```
210 INPUT A$ !Input can be numeric or string

220 IF A$ = "" THEN FG%(13%)=-1%

230 A = VAL(A$) !where a numeric value is needed
```

flg 14 - When set to 1 by the user, math errors are ignored and do not cause termination of the program

flg 15 - Automatically set to 1 when a math error occurs

The following routine can provide the same features as flg 14 and flg 15:

```
240 IF FG%(14%)=-1%AND D=0 THEN A=10^75
```

Dividing by Zero

```
250 A=100/D
260
```

```
240 IF FG%(15%)=-1%\GO TO 260
```

Square root of a negative number

```
240 IF FG%(14%)=-1%AND SGN (D)=-1
      THEN Y=-Y \ FG%(15%)=-1%
250 X=SQR(Y)
```

Square root of a negative number

```
240 IF FG%(14%)=-1%AND SGN(D)=-1
      THEN Y = -Y \ FG%(15%)=-1%
250 X=SQR(Y)
```

An Alternate Error Handling Routine

Program 3 - BASIC Version

```
850 ! 'ERROR',      ERROR HANDLER
860 ! VERSION 1.0, 27 NOVEMBER 1979
900 DIM FG%(15%) \ D=0 \ Y=-1 \ FG%(14%)=-1%      ! INITIALIZE
1000 ON ERROR GOTO 2100
1010 A=100/D                                      ! DEVIDE BY ZERO
1020 OFF ERROR \ PRINT A
1030 ON ERROR GOTO 2200
1040 X=SQR(Y) \ PRINT X                           ! SQUARE ROOT OF -1
1050 OFF ERROR
1060 GOTO 2500
2100 IF NOT FG%(14%) GOTO 2400
2105 IF ERR=603 THEN A=10^75 \ RESUME 1020    ! ERROR HANDLER
2200 IF ERR=604 THEN Y=-Y \ RESUME 1040       ! ERROR HANDLER
2400 PRINT "ERROR NUMBER",ERR
2500 END
```

Program 3 - HPL

```
0: "Error Handle
 r":
1: 0→D;-1→Y;sfg
 14
2: on err "error
 "
3: 100/D→A
4: prt A
5: on err "ERROR
 "
6: √Y→X;prt X
7: end
8: "error":if
 flg14;1e99→A;
 gto 4
9: stp
10: "ERROR":if
 flg14;-Y→Y;gto
 6
11: stp
12: end     FILE 6
*20321
```

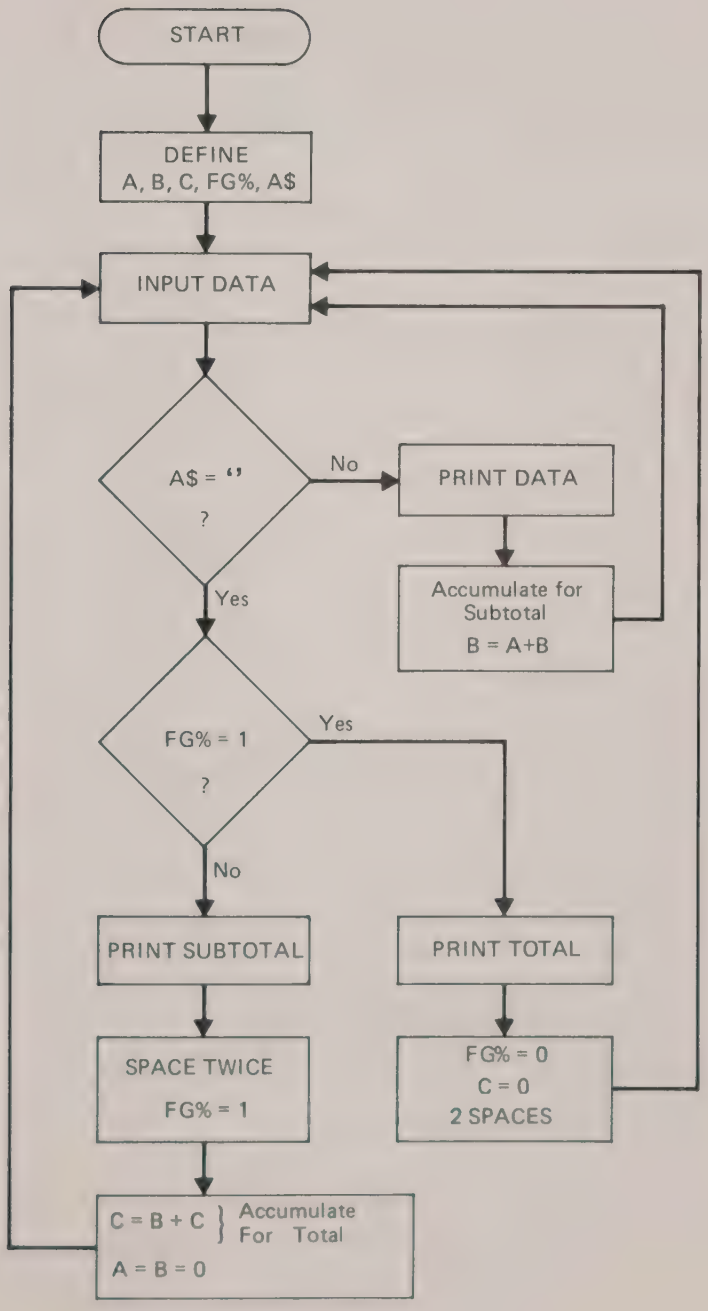
```
9.999999999e 99
1.00
```

The following program demonstrates the use of the null string in place of the HPL flg 13.

ADDING MACHINE USER INSTRUCTIONS

1. Key program in (Immediate or Edit mode)
2. Type: RUN ! Immediate mode
3. Press: RETURN
4. "ENTRY?" is displayed
5. Type: numerical entry
6. Press: RETURN
7. "ENTRY?" is displayed
8. If more entries are needed, go to Step 5
9. If subtotal is desired:
   a. Do not make an entry
   b. Press: RETURN
   c. SUBTOTAL is printed
10. "ENTRY?" is displayed
11. If more entries need to be made prior to another subtotal, go to Step 4
12. If a final total is needed:
   a. Do not make an entry
   b. Press: RETURN
   c. TOTAL is printed
13. Go to Step 4

ADDING MACHINE FLOW CHART



PROGRAM 4 - BASIC

```
1 ! 'ADD1',            ADDING MACHINE
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
30 ' ***************************************************************************
40 '
100 ' THIS PROGRAM DUPLICATES THE FUNCTION OF AN ADDING MACHINE, AND IT
110 ' DEMONSTRATES THE USE OF THE FLAG "FGZ".   "FGZ IS SET TO INDICATE
120 ' THAT A SUBTOTAL HAS BEEN TAKEN.  "FGZ" IS CLEARED AS SOON AS MORE
130 ' DATA IS ENTERED.  FGZ IS NOT CLEARED IF RETURN IS PRESSED WITHOUT
140 ' FIRST ENTERING MORE DATA; IN THIS CASE THE PROGRAM IS DIRECTED TO
150 ! TAKE A TOTAL.
160 !
170 '***************************************************************************
180 '
600 !                   VARIABLE DEFINITIONS
610 !
620 ' A$ RECEIVES THE DATA INPUT, AS A STRING.  A$ BECOMES A NULL STRING
625 ! WHEN RETURN IS PRESSED WITHOUT MAKING ANY DATA ENTRY.
630 ! A IS THE NUMERICAL VALUE OF A$.
640 ! B ACCUMULATES A FOR THE SUBTOTAL.
650 ! C ACCUMULATES B FOR THE TOTAL.
655 CU$ = CHR$(27%) + "[" + "A"       ! CURSOR UP ONE ROW
660 !
670 '***************************************************************************
680 !
690 !                    MAIN PROGRAM
700  PRINT CHR$(27%) + "[" + "0;2r";   ! RESET SCREEN
2000 A=0 \ B=0 \ C=0 \ FG%=0%     ! INITIALIZE
2010 PRINT "  ENTRY"; \ INPUT A$  ! DATA INPUT
2015 PRINT CU$;           ! CURSOR UP ONE ROW
2020 IF A$=""GOTO 2060    ' TEST FOR NO ENTRY (IF SO USER WANTS SUBTOTAL)
2030 A=VAL(A$)            ! RETRIEVE NUMERICAL VALUE OF THE ENTRY
2035 PRINT USING "##########.##", "        ";A
2040 FG%=0                ! CLEAR FLAG FROM LAST SUBTOTAL (DON'T TOTAL)
2050 B=A+B \ GOTO 2010    ! ACCUMULATE FOR SUBTOTAL
2060 IF FG% GOTO 2110     ' IF FLAG SET (FROM LAST SUBTOTAL) TAKE TOTAL
2070 PRINT CU$ \ PRINT USING "##########.##","SUBTOTAL";B
2075 PRINT                ! LINE FEED
2080 C=B+C \ A=0 \ B=0    ' ACCUMULATE FOR TOTAL, ZERO ENTRY & SUBTOTAL
2090 FG%=1                ! SET FLAG TO SHOW SUBTOTAL HAS BEEN TAKEN
2100 GOTO 2010            ' BACK FOR ANOTHER ENTRY (OR POSSIBLE SUBTOTAL)
2110 PRINT CU$ \ PRINT USING "##########.##","TOTAL***";C
2120 PRINT \ PRINT        ! LINE FEEDS
2130 FG%=0 \ C=0          ! CLEAR SUBTOTAL FLAG, ZERO OUT TOTAL
2140 GOTO 2010
2150 END
3000 !
3010 !
3020 '
3030 '
3040 '
3050 ' THE PROGRAM ON THE NEXT PAGE ILLUSTRATES HOW THIS
3060 ' IS PROGRAMMED IN HP!
3070 '
3080 '
3090 END
```

PROGRAM 4 - HPL VERSION: ADDING MACHINE

```
                20.00
                30.00
                50.00

SUBTOTAL       100.00


                50.00
                10.00
                40.00

SUBTOTAL       100.00


TOTAL          200.00
```

```
0: fxd 2;0→A→B→C
1: ent "Make
 Entry",A
2: if not flg13;
 prt A;cfg 1;A+
 B→B;gto 1
3: if flg1;prt
 "TOTAL";C;cfg
 1;spc 4;0→C;
 cfg 13;gto 1
4: spc ;prt "SUB
 TOTAL";B;spc 2;
 sfg 1;cfg 13
5: B+C→C;0→A→B;
 gto 1
6: end
*3410
```

FILE 11

3.10 COMMAND EQUIVALENTS

Table 3-4

The following table equates HPL commands with equivalent BASIC commands:

9825A HPL	1720A BASIC
cont (continue)	CONT
del (delete)	DELETE
erase	DELETE ALL
erase a	not available
erase v	not available
erase k	not needed
erase fn	not needed
fetch	EDIT
run	RUN
run <line No.>	not allowed
run <label>	not available
cont <line No.>*	GO TO <line No.>
not available	CONT TO <line No.>**
list	LIST

* Line No. is where program begins running

** Line No. is where program stops running; program begins at line following last executed statement when the next CONT command is executed

3.11 STANDARD STATEMENT EQUIVALENTS

All standard HPL statements are shown below. Exact equivalents in 1720A BASIC are given here. HPL statements which require more than one 1720A BASIC statement for conversion, are referred to another page.

Table 3-5

STANDARD STATEMENT EQUIVALENTS

HPL	1720A
beep	PRINT CHR$(7)
cfg (clear flag)	see Section 3.8
cmf (compliment flag)	see Section 3.8
csv (clear simple variables)	A=0\B=0\...\Z=0
deg (degrees)	see following page
dim (dimension)	DIM,see * and see following page
dsp (display)	PRINT
end	END
enp (enter-print)	INPUT (prints on display)
ent (enter)	INPUT (prints on display)
flt (float)	use PRINT USING statement with E format***
fxd (fixed)	described later in this section
grad	described later in this section
gsb (gosub)	GOSUB
gto (goto)	GO TO (also see example at end of this section)
if	IF, see items 3,5,22 in section 4.1 and see examples at end of this section
jmp (jump)	described at end of this section
list	LIST (CRT only)
lkd (live keyboard disable)	disconnect keyboard
lke (live keyboard enable)	connect keyboard
nor (normal)	TRACE OFF**
prt (print)	PRINT
rad (radian)	Angles are always specified in radians
ret (return)	RETURN
sfg (set flag)	See section 3.8
spc (space),i.e.line feed	PRINT or PRINT CHR$(32)

stp (stop)	STOP
trc (trace)	TRACE ON
units	Must be programmed
wait	WAIT

* close equivalent
** cannot specify line numbers; turns all tracing off
***see example of PRINT USING with E format on Line 125 of Trigonometric Table program in Section 3.16

deg All angular calculations are handled in radians by the 1720A. Angles entered in degrees must be converted to radians. If output is desired in degrees, then radians must be converted to degrees. Conversion is given below. A full discussion on degrees and radians is given in the appendix of this document.

Radians = Degrees/57.295795130823
Degrees = Radians*57.295795130823

Note: $180/\pi$ =57.2957795130823

dim The 1720A BASIC equivalent is DIM. Round brackets are used around the dimensions. All dimensions start at 0. Also refer to Table 3-3a.

Examples:

	9825A HPL	1720A BASIC
1.	dim A[10] results in 10 variables, A[1] thru A[10]	DIM A(10%) results in 11 variables, A(0%) thru A (10%)
2.	dim A, B, C	not allowed
3.	dim Y[-3:0,4:6],I,J	not allowed, however, a simple conversion program could be used:
	20:dim Y[-3:0,4:6],I,J	<line No.>DIM Y(4%,3%)\ I%=0\J%=0
	21:ent I,J	<line No.> INPUT I%,J%
	22. Y[I,J]=π R↑2	<LINE NO.>.Y(I%+3%,J%-4%)=PI*R^2
4.	DIM Y [10,5,5,4]	Only two dimensions are allowed. See *Section 3.7* for a conversion program to handle more than two dimensions

fxd The PRINT USING statement can be used in 1720A BASIC to handle fixed notation. Examples follow:

```
                9825A HPL                      1720A BASIC
                ----- ---                      -----

1.   0: ent N                          INPUT N
     1: fxd 0; prt N                   PRINT USING
                                         "S##############",N
     2: fxd 1; prt N                   PRINT USING
                                         "S##############.#",N
     3: fxd 2; prt N                   PRINT USING
                                         "S##############.##",N
     4: fxd 3; prt N                   PRINT USING
                                         "S##############.###",N
     5: end
        125  e  10--->N                N=125  E  10


     result 1250000000000              result 1250000000000
            1250000000000.0                   1250000000000.0
            1.250000000e 12                   1250000000000.00
            1.250000000e 12                   1250000000000.000
```

grad All angular calculations are handled in radians by the 1720A. Angles entered in grads must be converted to radians. If output is desired in grads, then radians must be converted to grads. A full discussion on grads is given in this appendix of this document. Conversion is given below:

grads = radians * 63.6619772367581

radians = grads/63.6619772367581

note: $63.6619772367581 = \frac{400}{2\pi}$

```
if              9825A HPL                      1720A
                ----- ---                      -----

14:if X=Y;gto 50                   <no.> IF X=Y GO TO 50
15:if Z; Z-1-->Z; print Z          <no.> IF Z THEN Z=Z-1 \
                                     PRINT Z
```

Note: Read PROGRAMMING PITFALLS, Section 4.1, items 3, 5, and 22 as they apply to IF statements

9825A - 16:if X<1000;gto 100; if X<500;gto 110;if X<100;gto 130

```
1720A - 16 IF X<1000 THEN IF X<500 THEN IF X<100
        THEN 130 ELSE 17 ELSE 100 ELSE 110

                        OR

        16 IF X<100 GOTO 130
        17 IF X<500 GOTO 110
        18 IF X<1000 GOTO 100

                        OR

        16 X=1-((X<1000) + (X<500) + (X<100)
        17 ON X GOTO 18, 100, 110, 120
```

jmp	9825A HPL	1720A
	40:prt B; jmp (B+1-->B)=10	<No.>FOR B%=1% to 10% <No.>PRINT B% <No.>NEXT B%
	40:jmp 3X+10	40 ON X GO TO 53,56, 59,62,65
	Note: limits of X must be known: for example assume 1<=X<=5	

3.12 STANDARD FUNCTION EQUIVALENTS

All standard HPL functions are shown below. Exact equivalents in 1720A BASIC are given here. HPL functions which require more than one 1720A BASIC statement for conversion, are referred to on another page.

Table 3-6

Standard Function Equivalents

HPL	1720A
abs(absolute value)	ABS (X)
acs (arc cosine)	see below
asn (arc Sine)	see below
atn (arc tan)	ATN (X)
cos (cosine)	COS (X)
drnd (digit round)	see next page
exp (exponential)	EXP (X)
flg (flag)	see section 3.9
frc (fraction)	X-INT (X)see next page
int (integer)	INT (X)see next page
log (commonlogarithm)	LOG (X)
ln (natural logarithm)	LN (X)
max (maximum)	see example, later in this section
min (minimum)	see example, later in this section
prnd (power round)	see next page

ln (natural logarithm)	LN (X)
max (maximum)	see example, later in this section
min (minimum)	see example, later in this section
prnd (power round)	see next page
res	printed in display
rnd	RND**
sgn	SGN (X)
sin	SIN (X)
√‾	SQR (X)
tan	TAN (X)
tn	10^X

INT (X) is limited to integers within the range of -32768 to +32767. An alternate way of performing this function with strings (bypassing this limitation) is shown later in this section.

Defined functions are used in the solutions below to develop equivalent functions. Once a defined function statement has been executed, it need not be executed again in the same
---
program in order to use the function again and again.

When X=SIN(Y) <line No.> DEF FNC(X)=ATN(SQR(1-X^2)/X)

ARC COS(Y) or acs (Y) in 1720A BASIC becomes FNC(X) *

When X=COS(Y) <line No.> DEF FNS(X)=ATN(X/SQR(1-X^2))

ARC SIN(Y) or asn(Y) in 1720A BASIC becomes FNS(X) *

<line No.> DEF FN F(X)=X-INT (X)

frc in 1720A BASIC becomes FNF (X)

* See proof in Appendix, section 5F,3

** <line No.>RANDOMIZE ! This statement uses the clock value as a seed to initialize the random value generator RND

Program 5

```
10  ! POWER ROUNDING, 'PRND',                JULY 24, 1979
20 PRINT"NO. TO BE POWER ROUNDED, POWER"; ! PROMPT
30 INPUT X,P%                             ! INPUT NUMBER
40 X=INT(.5+X/10^P%)*10^P%                ! ROUND NUMBER
50 PRINT X                                ! PRINT ROUNDED NUMBER
60 GOTO 20
70 END
```

Program 6

```
10 ! DIGIT ROUNDING, 'DRND',              JULY 24, 1979
20 PRINT"NO. TO BE DIGIT ROUNDED, DIGITS";  ! PROMPT
30 INPUT X,Q%                               ! INPUT NUMBER
40 Y$=NUM$(X)                               ! STRING NUMBER
50 K%=INSTR(2%,Y$,".")                      ! FIND POSITION OF DECIMAL
60 K%=K%-(LEN(Y$))*(K%=0%)                  ! CHECK FOR IMPLIED DECIMAL
70 P%=K%-Q%-2%                              ! CONVERT TO POWER ROUND
80 X=INT(.5+X/10^P%)*10^P%                  ! POWER ROUND NUMBER
90 PRINT X                                  ! PRINTROUNDED NUMBER
100 GOTO 20
110 END
```

Program 7

```
100 ! INTEGER AND FRACTIONAL PARTS USING STRINGS, 'INTFRC', JULY 24, 1979
110 PRINT 'INPUT NUMBER'                          ! PROMPT
120 INPUT Y$                                      ! INPUT NUMBER
130 K%=INSTR(1%,Y$,".")                           ! POSITION OF DECIMAL
140 I$=LEFT(Y$,K%-1%)                             ! INTEGER PART
150 I=VAL(I$)                                     ! UNSTRING
160 PRINT I
170 F$=RIGHT(Y$,K%)                               ! FRACTIONAL PART
180 F=VAL(F$)                                     ! VALUE
190 PRINT F
200 GOTO 100
210 END
```

```
<line No.> DEF FNI(X)=
      VAL (LEFT(NUM$(X),INSTR (1%,NUM$(X),".")-1%))
where FNI(X) replaces INT(X)

<line No.> DEF FNF(X)=
      VAL (RIGHT(NUM$(X)<INSTR(1%,NUM$(X),".")))
where FNF(X) replaces frc(X)
```

Program 8

```
10 ! FIND MAX AND MIN VALUES, 'MAXMIN',          JUNE 1, 1979
20 DIM A(8%,2%)                          ! DIMENSION EXAMPLE ARRAY
30 FOR I%=0% TO 8%
40 FOR J%=0% TO 2%
50 A(I%,J%)=100*RND                      ! ASSIGN RANDOM NUMBER
60 NEXT J%
70 NEXT I%
80 MN=A(0%,0%) \ MX=A(0%,0%)             ! INITIALIZE MIN AND MAX
90 FOR I%=0% TO 8%
100 PRINT A(I%,0%), A(I%,1%), A(I%,2%)   ! PRINT ARRAY ELEMENTS
110 FOR J%=0% TO 2%
120 IF A(I%,J%)>MX THEN MX=A(I%,J%)      ! COMPARE FOR MAX VALUE
130 IF A(I%,J%)<MN THEN MN=A(I%,J%)      ! COMPARE FOR MIN VALUE
140 NEXT J%
150 NEXT I%
160 PRINT                                ! LINE FEED
170 PRINT "MAX =",MX, "MIN =",MN
180 END
```

3.13 TAPE CARTRIDGE STATEMENT EQUIVALENTS

The Fluke 1720A is equipped with a flexible disk (Electronic Disk available as an option; uses the same statements and is 4 to 40 times faster than the floppy) rather than the slower tape cartridge used by the 9825A. The following table shows the disk commands which may be used in place of the tape commands in your program conversion. Example conversions are given in programs 9,10,11 and 12 in this section. The 1720A Programming Manual contains detailed descriptions of the use of these disk statements. File names of 6 or less characters (3 character extension is optional) are used in place of the HPL file number.

Table 3-7

Tape Cartridge Statement Equivalents

9825A HPL	1720A BASIC
avd (auto verify disable) ave (auto verify enable)	Not available; however, downloading of the disk into the 1720A is verified by a checksum
ert (erase tape)	/Z !This is zeroing the directory*
fdf (find file)	Not needed with file by name system on disk
idf (identify file)	Not needed with file by name system on disk
ldb (load binary program)	All system interpretors are loaded in binary via the File Utility Program (FUP)
ldf (load file program)	OLD<file name>* ! file name must be in quotes or be a string
ldf (load file - data)	INPUT <list of variables separated by commas> ! must be preceded by OPEN** statement.
ldk (load keys)	special function keys are available through "Touch Sense" and must be part of the current program
ldm (load memory)	Not available,however, battery backup for Electronic Disk is available
ldp (load program)	RUN<file name>* ! file name in quotes or a string

mrk (mark files)	/F !format a diskette*
rcf (record file - program	SAVE <file name>* ! file name in quotes or a string
rcf (record file - data)	PRINT #<file No.>, <list of variables separated by commas> Note: commas must be recorded as "," (see line 2070 in Program 11 in this section. PRINT #<file No.>, USING <string> <list> / format
rck (record keys)	special function keys are available through "Touch-Sense" and must be part of the current program. See Section 4.2.1
rcm (record memory)	not available,however, battery backup for Electronic Disk is available
rew (rewind)	not needed by disc
tlist (tape list)	/L ! List the file directory*
trk (track)	tracks are handled by the system
vfy (verify)	write operations to the disk and read operations from the disk to the user memory are verified by a checksum.

*** Not available in BASIC; available in File Utility Program (FUP) only.

** Refer to Appendix, 5B for discussion of File Nos., channel Nos., device Nos. and the OPEN statement.

* Fluke BASIC does not support the recording of sections of programs. This can be accomplished, however, by first deleting unwanted lines (DELETE<line No.>-<line No.>) and then recording the program (SAVE 'file name').

Fluke BASIC does not support the merging of separately recorded programs into one contiguous program. Separate programs may, however, be merged by using the File Utility Program (FUP) via the Merge Utility (<file name>=<file name>,<file name><CR>
/
final merged program

Program 9 - Basic Version

```
1 ! 'SDATA', DATA HANDLER 1
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
100  ! THIS PROGRAM DEMONSTRATES HOW DATA CAN BE STORED ON THE FLOPPY
110  ! WHICH WILL REPLACE THE CASSETTE IN YOUR PROGRAMS.  DATA IS
120  ! STORED ONE ITEM PER LINE ONTO THE DISK.  THIS WAY ONE DATA ITEM
122  ' COULD OCCUPY AN ENTIRE LINE, IF NECESSARY.  THEN DATA IS INPUT
130  ! FROM THE DISK, ONE ITEM AT A TIME.  BE SURE TO LOOK AT RAY.DAT
135  ! IN FUP.
140  !
2000 CLOSE 5,6                              ! MAKE SURE FILES ARE CLOSED
2010 OPEN 'RAY.DAT' AS NEW FILE 5 SIZE 1    ' OPEN CHANNEL FOR DATA STORAGE
2020 PRINT"ENTER DATA";                     ! PROMPT
2030 INPUT A$                               ' INPUT DATA FROM KEYBOARD
2040 IF A$="END" THEN 2070                  ! INPUT "END" WHEN THROUGH
2050 PRINT#5,A$                             ' STORE DATA
2060 GOTO 2020                              ! GO GET NEXT DATA INPUT
2070 CLOSE 5 \ PRINT                        ! REPOSITION DATA POINTER
2080 OPEN 'RAY.DAT' AS OLD FILE 6 SIZE 1    ! OPEN CHANNEL TO RECEIVE DATA
2090 ON ERROR GOTO 2130                     ' DETECTS END OF FILE
2100 INPUT#6,A$                             ' INPUT DATA
2110 PRINT A$                               ! DISPLAY DATA AS INPUT
2120 GOTO 2100                              ' GO GET NEXT DATA
2130 CLOSE 6                                ' CLOSE CHANNEL
2140 END
```

Program 10 - Basic Version

```
1 ! 'VDATA', DATA HANDLER 1A
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
100  ' THIS PROGRAM DEMONSTRATES HOW  ALL THE DATA CAN BE PRINTED ONTO
120  ' THE DISK AS A SINGLE LINE.  THEN DATA IS INPUT FROM THE DISK AS
125  ' AN ENTIRE ARRAY. DON'T MAKE YOUR NUMBERS TOO LARGE OR YOU WILL
130  ! EXCEED THE LENGTH OF A LINE.  BE SURE TO LOOK AT THE FILE
140  ! PAT.DAT IN FUP
150  !
2000 CLOSE 2                                          ! MAKE SURE FILE IS CLOSED
2010 DIM C(10%)
2020 OPEN "PAT.DAT" AS NEW FILE 2 SIZE 1              ! OPEN CHANNEL
2030 FOR I%=0% TO 10%
2040 INPUT C(I%)                                      ! INPUT DATA
2050 PRINT #2, C(I%);                                 ' STORE DATA
2060 IF I% <> 10% THEN PRINT #2, ',"; ELSE PRINT #2   ! INSERT DELIMITERS
2070 NEXT I%
2080 CLOSE 2                                          ! REPOSITION DATA POINTER
2090 DIM B(10)
2100 OPEN "PAT.DAT" AS OLD FILE 2 SIZE 1              ! OPEN CHANNEL
2110 INPUT #2, B(0..10)
2120                                                  ' INPUT DATA
2130 PRINT B(0..10)
2140                                                  ! DISPLAY DATA
2150 CLOSE 2                                          ' CLOSE CHANNEL
2160 END
```

Program 11 - Basic Version

```
1 ! 'SPRT', DATA HANDLER 2, SERIAL PRINTING TO DISK
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
100 ! THIS PROGRAM DEMONSTRATES A COMBINATION OF WAYS TO PRINT DATA ONTO
110 ! THE DISK.  NOTE HOW THE PRINT STATEMENTES CORRELATE WITH THE
120 ! INPUT STATEMENTS.  BE SURE TO LOOK AT DATA2 .DAT IN FUP.
130 !
2000 CLOSE 3                                  ! MAKE SURE CHANNEL IS CLOSED
2010 OPEN 'MFO:DATA2.DAT'AS NEW FILE 3 SIZE 1 ! OPEN CHANNEL
2020 A$="THIS IS A TEST" \ A1=PI \ A2=2*PI    ! GENERATE DATA
2030 DIM B(10%)                               ! DIMENSION ARRAY FOR MORE DATA
2040 FOR I%=0% TO 10%
2050 B(I%)=I%+10%                             ! GENERATE MORE DATA
2060 NEXT I%
2070 PRINT #3, A$;",";A1;",";A2               ! STORE DATA
2080 A$=SPACE$(15%) \ A1=0 \ A2=0             ! RE-INITIALIZE
2090 FOR I%=0% TO 10%
2100 PRINT #3, B(I%) \ B(I%)=0                ! STORE OTHER DATA
2110 NEXT I%
2120 CLOSE 3                                  ! REPOSITION DATA POINTER
2130 OPEN "MFO:DATA2.DAT" AS OLD FILE 3 SIZE 1    ! OPEN CHANNEL
2140 DIM C(10%)                               ! ALTERNATE ARRAY
2150 INPUT #3, B$,B1,B2                       ! INPUT DATA
2160 FOR I% = 0% TO 10%
2170 INPUT #3, C(I%) \ PRINT C(I%)            ! INPUT OTHER DATA
2180 NEXT I%
2190 PRINT B$,B1,B2                           ! DISPLAY DATA
2200 CLOSE 3                                  ! CLOSE CHANNEL
2210 END
```

Program 12 - Basic Version

COMPARE

```
1 ! 'PDATA', DATA HANDLER 3
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
100 ! THIS PROGRAM DEMONSTRATES ANOTHER WAY OF DOING THE SAME THING AS
110 ! THE 'SPRT' PROGRAM ABOVE.  NOTE HOW LINE 2190 IS USED TO MAKE THE
120 ! DISPLAY IDENTICAL TO THE 'SPRT' PROGRAM.  BE SURE TO LOOK AT
130 ! DATA3 .DAT IN FUP.
140 !
2000 CLOSE 3                                  ! MAKE SURE CHANNEL IS CLOSED
2010 OPEN 'MFO:DATA3.DAT'AS NEW FILE 3 SIZE 1     ! OPEN CHANNEL
2020 A$="THIS IS A TEST" \ A1=PI \ A2=2*PI    ! GENERATE DATA
2030 DIM B(10%)                               ! DIMENSION ARRAY FOR MORE DATA
2040 FOR I%=0% TO 10%
2050 B(I%)=I%+10%                             ! GENERATE MORE DATA
2060 NEXT I%
2070 PRINT #3, A$ \ PRINT #3, A1 \ PRINT #3, A2 ! STORE DATA
2080 A$=SPACE$(15%) \ A1=0 \ A2=0             ! RE-INITIALIZE
2090 FOR I%=0% TO 10%
2100 PRINT #3, B(I%) \ B(I%)=0%               ! STORE OTHER DATA
2110 NEXT I%
2120 CLOSE 3                                  ! REPOSITION POINTER
2130 OPEN "MFO:DATA3.DAT" AS OLD FILE 3 SIZE 1    ! OPEN CHANNEL
2140 DIM C(10%)                               ! ALTERNATE ARRAY
2150 INPUT #3, B$ \ INPUT #3, B1 \ INPUT #3, B2  ! INPUT DATA
2160 FOR I% = 0% TO 10%
2170 INPUT #3, C(I%) \ PRINT C(I%)            ! INPUT OTHER DATA
2180 NEXT I%
2190 PRINT B$,B1,B2                           ! DISPLAY DATA
2200 CLOSE 3                                  ! CLOSE CHANNEL
2210 END
```

Program 12 - HPL Version

BEFORE

```
trk  1
#0
 0    0       1000
#1
 0    0       1000
#2
 0    0       1000
#3
 0    0       1000
#4
 0    0       1000
#5
 0    0       1000
#6
 3    128     500
#7
 3    128     500
#8
 0    0       0
```

```
          10.00
          11.00
          12.00
          13.00
          14.00
          15.00
          16.00
          17.00
          18.00
          19.00
          20.00
THIS IS A TEST
           3.14
           6.28
STORED IN FILE
           8.20
```

AFTER

```
trk  1
#0
 0    0       1000
#1
 0    0       1000
#2
 0    0       1000
#3
 0    0       1000
#4
 0    0       1000
#5
 0    0       1000
#6
 3    128     500
#7
 3    128     500
#8
 3    128     500
#9
 0    0       0
```

```
 0: "DATA HANDLER
  ":
 1: "June 6, 1979
  ":dim A$[15],
  A[2],B[0:10];
  trk 1;rew;-1→A
 2: A+1→A;trk 1;
  fdf A;idf A,B,
  C,D,E;gto +0;
  if D=0;fdf A;
  mrk 1,500;gto +
  1
 3: "THIS IS A
  TEST"→A$;π→A[1]
  ;2π→A[2]
 4: dim B$[15]
 5: for I=0 to 10
 6: I+10→B[I]
 7: next I
 8: rcf A,A$,A[*
  ],B[*]
 9: "RETRIEVE
  DATA":
10: "filler":
11: "filler":
12: "filler":
13: "filler":
14: dim F$[15],
  G[2],H[0:10]
15: ent "READY?"
  ,B$;gto +0;if
  B$="YES";gto +1
16: trk 1;ldf A,
  F$,G[*],H[*]
17: for I=0 to
  10
18: prt B[I]
19: next I
20: prt A$
21: prt A[1],
  A[2]
22: prt "STORED
  IN FILE";A
23: end
*16581
```

FILE 0

3.14 STRING ROM STATEMENT EQUIVALENTS

Table 3-8

String ROM Statement Equivalents

9825A HPL		1720A BASIC
dim(dimension)	dimA$[50],B$[50] dim A$[N,C] N strings of C characters	A$=""\B$="" DIM A$(N%)
dsp(display)	dsp X,C$,D$[N]	PRINT X,C$,D$(N%)
enp(enter-print)	enp"size",S$	PRINT "SIZE";\INPUT S$
ent(enter)	ent "COLOR",C$	PRINT "COLOR";\INPUT C$
if	if X$ = Y$	IF IF X$=Y$ THEN ... (or IF X$=Y$ GOTO...)
prt(print)	prt X$, Y$, X, Y	PRINT X$,Y$,X,Y
ldf(loadfile)	ldf(file No.)N, X$,Y$, X, Y	INPUT #C (internal Channel No.), X$, Y$,X,Y
rcf(record file)	rcf N,X$,Y$,X,Y	PRINT #C(internal channel No.), X$,Y$,X,Y
--->	X$--->Y$;"NO"--->Z$	= Y$=X$ \ Z$="NO"
len(length)	len (X$)--->Y	LEN Y%=LEN(X$)
pos(position)	pos(X$[3],".")-->Y pos(X$[N], Y$)-->Y	INSTR *Y%=INSTR(3%,X$,".") Y%=INSTR(N%,X$,Y$) / \ start at position look for

Note: The HPL file number will have been replaced with a 1720A disk "NAME"; then this "NAME" must have been assigned a channel No. via an OPEN statement (see page 5B in Appendix).

val(value) val (X$)-->Y	VAL Y%=VAL (A$)
&(concatination) X$&Y$--->X$	+ X$=X$+Y$
num(ASCII number)num(X$[2,2])-->Y	ASCII Y%=ASCII (MID(X$,2%, 1%)) **
str(string) str(X)-->X$; str(3.172)--->Y$	NUM X$=NUM$(X)\Y$=NUM$(3.172)
char(character) char (51)--->B$	CHR$ B$=CHR$(51%)
cap(capital) cap (B$)--->B$	B$=CHR$(ASCII(B$)-32%)

* Returns sub string position

** X$[2,2]--->A$ in HPL becomes A$=MID(X$,2%,1%)

9825A HPL	1720A BASIC
"?"--->A$[3,3]	A$=LEFT(A$,2%)+"?"+RIGHT(A$,4%)

If you wanted to pick out the state and zip code from the following string:

```
A$ = "I    L  I  V  E    I  N    W  A  S  H  I  N  G  T  O  N
      1  2 3  4  5  6  7 8  9 10 11 12 12 14 15 16 17 18 19 20

       A  N  D    M  Y     Z  I  P    C  O  D  E    I  S
      22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

       9   8   2   0   4"
      41  42  43  44  45
```

It might be done in HPL like this:

```
A$[11,20]&A$[40,45]--->B$
```

It might be done in 1720A BASIC like this:

```
B$=MID(A$,11%,10%)+RIGHT(A$,40%)

B$="W  A  S  H  I  N  G  T  O  N     9  8  2  0  4"
    1  2  3  4  5  6  7  8  9 10  11 12 13 14 15 16
```

	HPL Equivalent
Y$ = SPACE$(N%)	
String of N% spaces	"b"---> Y$[1,N] where "b" is a space

Y$ = TAB(N%)

Returns a string of spaces. Its length depends on the current position of the printhead and the desired position.

Y$=CPOS(M%,N%)

Returns a character string consisting of an escape sequence to position the cursor at row (vertical position) M% and column (horizontal position) N%.

Y$=NUM$(X,A$)

String of numerics showing value of X as PRINT using A$,X would print it; for example, NUM$(-50,'S#.##^^^^') would return "-5.00E+01".

3.15 ADVANCED PROGRAMMING ROM STATEMENT EQUIVALENTS

The following table gives BASIC equivalents for the HPL Advanced Programming ROM:

Table 3-9

Advanced Programming ROM Statement Equivalents

9825A	1720A BASIC
STATEMENTS	
for	FOR
next	NEXT
by	STEP
xref (cross reference of variables and line numbers)	Use TRACE statement to trace variables and their line numbers
SUBROUTINE SUBPROGRAMS	
cll(call) cll'drop'(A,B,C)	Use standard subroutine; passing variables by assignment to a temporary variable
FUNCTION SUBPROGRAM	
'STD'(A,B,C)	Use defined function (see examples in section 3.12)
p-numbers	See cll, above
FUNCTIONS	
fts (full to split)	See program 13 in this section
stf (split to full)	See explanation at end of this section
fti (full to integer)	See program 14 in this section
itf (integer to full)	See explanation at end of this section

Program 13
Basic Version

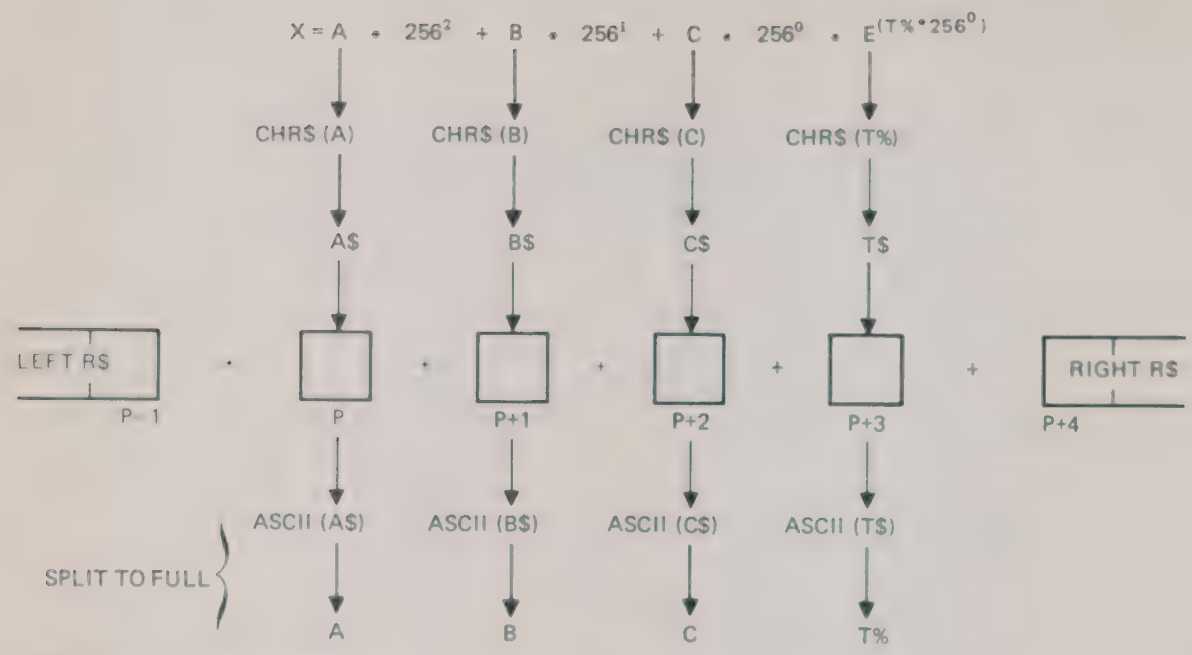
```
1 ! 'FTSSTF', FULL TO SPLIT-SPLIT TO FULL
7 ! VERSION 1.0, 20 NOVEMBER 1979
20 !
100 !                    FULL TO STRING
120 !  THIS PORTION OF THE PROGRAM TAKES A NUMBER AND USING BASE 256 BY
130 !  EQUATING THIS WITH THE ASCII CHARACTER NUMBERS, STORES THE NUMBER
140 !  AS A STRING.  THIS PROGRAM IS EQUIVALENT TO THE HPL FUNCTION "fts"
150 !  AND ALLOWS THE SAME SAVINGS IN BYTES OF STORAGE SPACE.  YOU MUST
160 !  ALLOW 4 POSITIONS OF STORAGE FOR EACH NUMBER, E.G., IF YOU STORE A
170 !  NUMBER AT POSITION 8, THIS NUMBER OCCUPIES POSITIONS 8 THROUGH 11.
180 !  AS WITH "fts" THERE IS A LIMIT OF 6 SIGNIFICANT DIGITS.
190 !
2000 DEF FNI(Z)=VAL(LEFT(NUM$(Z),INSTR(1%,NUM$(Z),".")-1%)) !INTEGER FUNCTION
2010 DIM FG%(1%)                              ! DIMENSION FLAGS
2020 FG%(0%)=0% \ FG%(1%)=0%                  ! INITIALIZE FLAGS
2030 R$=SPACE$(500%)                          ! INITIALIZE STRING STORAGE
2040 PRINT "ENTER NUMBER";                    ! PROMPT
2050 INPUT X
2060 IF X<0 THEN FG%(0%)=1% \ X=-X            ! NEGATIVE NO. HANDLER(SEE LINE160)
2070 IF X<>0 THEN D%=INT(LOG(X))              ! HOW BIG A NUMBER IS IT?
2080 T%=-(5%-D%)                              ! COMPUTE EXPONENT
2090 X=X*10^(5-D%) \ X=FNI(.5+X)              ! FORCE 6 SIGNIFCANT DIGITS & ROUND
2100 IF T%<0% THEN FG%(1%)=1% \ T%=-T%+128%   ! NEGATIVE EXPONENT HANDLER
2110 A=INT(X/256^2)                           ! MOST SIGNIFICANT DIGIT, BASE 256
2120 B=INT((X-A*256^2)/256)                   ! 2ND MOST SIG. DIGIT, BASE 256
2130 C=X-(A*256^2)-(B*256)                    ! 3RD MOST SIG. DIGIT, BASE 256
2140 IF FG%(0%) THEN A=A+128                  ! NEGATIVE NO. HANDLER(SEE LINE 70)
2150 A$=CHR$(A)                               ! EQUIVALENT ASCII CHARACTER
2160 B$=CHR$(B)                               ! EQUIVALENT ASCII CHARACTER
2170 C$=CHR$(C)                               ! EQUIVALENT ASCII CHARACTER
2180 T$=CHR$(T%)                              ! EQUIVALENT ASCII CHARACTER
2190 PRINT"ENTER POSITION OF STORAGE";        ! PROMPT
2200 INPUT P%
2210 R$=LEFT(R$,P%-1%)+A$+B$+C$+T$+RIGHT(R$,P%+4%)!STORE EQUIVALENT CHARACTERS
2220 PRINT MID(R$,P%,4%)                      ! VERIFY STORAGE
2230 !
2232 !                   STRING TO FULL
2234 !  THIS PORTION OF THE PROGRAM TAKES THE NUMBER WHICH HAS BEEN STORED AS
2235 !  A STRING IN THE FULL TO STRING PROGRAM, AND CONVERTS IT BACK TO A
2236 !  BASE 10 NUMBER, JUST AS "stf" IN HPL.  YOU MUST INPUT THE SAME STORAGE
2237 !  POSITION USED IN THE FULL TO STRING PROGRAM OR YOU WILL GET GARBAGE.
2238 !
2240 FG%(0%)=0% \ FG%(1%)=0%                  ! INITIALIZE FLAGS
2250 PRINT "ENTER POSITION OF STORAGE";        ! PROMPT
2260 INPUT P%
2270 A$=MID(R$,P%,1%)                         ! GET CHARACTER OF MOST SIG. DIGIT
2280 B$=MID(R$,P%+1%,1%)                      ! GET CHARACTER OF 2ND SIG. DIGIT
2290 C$=MID(R$,P%+2%,1%)                      ! GET CHARACTER OF 3RD SIG. DIGIT
2300 T$=MID(R$,P%+3%,1%)                      ! GET CHARACTER OF EXPONENT
2310 A=ASCII(A$)                              ! MOST SIG. DIGIT, BASE 256
2320 B=ASCII(B$)                              ! 2ND SIG. DIGIT, BASE 256
2330 C=ASCII(C$)                              ! 3RD SIG. DIGIT, BASE 256
2340 IF A>=128 THEN A=(A-128) \ FG%(0%)=1%    ! NEGATIVE NO. HANDLER
2350 T%=ASCII(T$)                             ! EXPONENT, BASE 256
2360 IF T%>=128% THEN T%=-(T%-128%)           ! NEGATIVE EXPONENT HANDLER
2370 X=(A*256^2+B*256+C)*10^T%                ! BACK TO BASE 10
2380 IF FG%(0%) THEN X=-X                     ! NEGATIVE NO. HANDLER
2390 PRINT USING"S#.#####^^^^",X              ! VERIFY NUMBER
2400 GOTO 2020
2410 END
```

Program 13 - HPL Version

```
0: "FULL TO STRI
 NG":
1: "JUNE 12,1979
  :
2: "THIS IS NOW
 A SUBROUTINE":
3: "FLAGS ALREAD
 Y USED":dim
 A$[15],B$[15],
 C$[15],T$[15],
 Y$[15]
4: "see line
 no. 7":
5: dim R$[500];
  " "→R$[1,500]
6: ent "ENTER
 NUMBER",X
7: cfg 0,1
8: if X<0;sfg 0;
 -X→X
9: if X#0;int(lo
 g(X))→D
10: -(5-D)→T
11: X*10↑(5-D)→X
 :str$(X)[1,6]→X
12: if T<0;sfg
 1;-T+128→T
13: int(X/256↑2)
 →A
14: int(Xmod256↑
 2/256)→B
15: X-A256↑2-
 B256→C
16: if flg0;A+
 128→A
17: char(A)→A$
18: char(B)→B$
19: char(C)→C$
20: char(T)→T$
21: ent "ENTER
 POSITION OF
 STORAGE",P
22: A$[1,1]→R$[P
 ,P];B$[1,1]→R$[
 P+1,P+1];C$[1,
 1]→R$[P+2,P+2]
23: T$[1,1]→R$[P
 +3,P+3]
24: prt R$[P,P+
 3]
25: "STRING TO
 FULL":
26: cfg 0,1
27: ent "ENTER
 POSITION OF
 STORAGE",P
28: "":
29: R$[P,P]→A$
30: R$[P+1,P+
 1]→B$
31: R$[P+2,P+
 2]→C$
32: R$[P+3,P+
 3]→T$
33: num(A$)→A
34: num(B$)→B
35: num(C$)→C
36: num(T$)→T
37: if A>=128;A-
 128→A;sfg 0
38: if T>=128;-
 (T-128)→T
39: (A256↑2+B256
 +C)*10↑T→X
40: if flg0;-X→X
41: fmt 1,e12.5;
 wrt 16.1,X
42: gto 6
43: end
*9:35
```

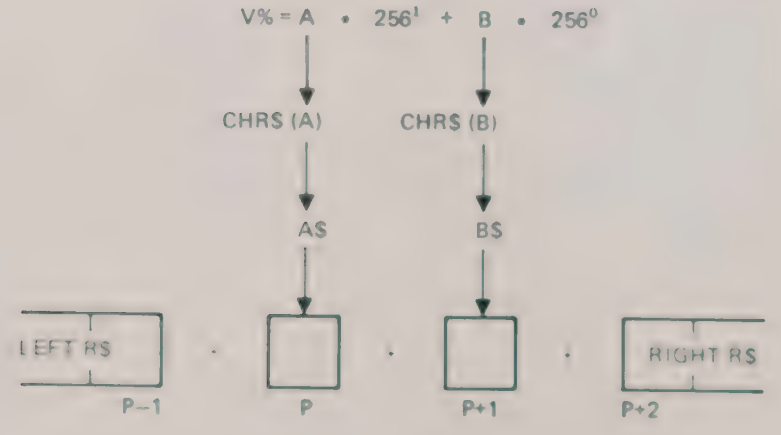
FILE 7

```
1 ! 'FTIITF', FULL TO INTEGER & INTEGER TO FULL
7 ! VERSION 1.0, 20 NOVEMBER 1979
10 !
20 !                          FULL TO INTEGER
30 !
100 ! THIS PROGRAM CONVERTS AN INTEGER IN THE RANGE OF -32766 TO 32767
200 ! AND CONVERTS IT TO A STRING USING BASE 256 WITH THE ASCII CHARACTER
210 ! CHARACTER SET.  THIS PROGRAM IS EQUIVALENT TO THE HPL FUNCTION "fti"
220 ! INCLUDING THE RESULTING SAVINGS IN SPACE USED FOR DATA STORAGE.
230 !
2000 FG%=0%                                        ! INITIALIZE FLAG
2010 PRINT "INTEGER TO BE STORED";                 ! PROMPT
2020 INPUT V%
2030 IF V%<0% THEN FG%=1% \ V%=-V%                 ! NEGATIVE NO. HANDLER
2040 A%=INT(V%/256%) \ B%=V%-256%*A%               ! V% MOD 256%
2050 IF FG% THEN A%=A%+128%                        ! REST OF NEGATIVE NO. HANDLER
2060 A$=CHR$(A%)                                   ! CONVERT TO ASCII CHARACTER
2070 B$=CHR$(B%)                                   ! CONVERT TO ASCII CHARACTER
2080 PRINT "ENTER POSITION OF STORAGE";            ! PROMPT
2090 INPUT P%                                      ! INPUT STORAGE POSITION
2100 R$=SPACE$(500%)                               ! INITIALIZE STORAGE STRING
2110 R$=LEFT(R$,P%-1%)+A$+B$+RIGHT(R$,P%+2%)       ! STORE INTEGER IN R$
2120 PRINT MID (R$,P%,2%)                          ! VERIFY STORAGE CHARACTERS
2121 !
2122 !                        INTEGER TO FULL
2123 !
2130 ! THE FOLLOWING PROGRAM RETRIEVES THE INTEGER FROM STORAGE AND IS
2132 ! EQUIVALENT TO THE HPL FUNCTION "fti".  YOU MUST USE THE SAME POSITION
2134 ! OF STORAGE ORIGINALLY ASSIGNED OR YOU WILL GET GARBAGE.
2136 !
2140 PRINT"ENTER POSITION OF STORAGE";              ! PROMPT
2150 INPUT P%                                      ! INPUT STORAGE POSITION
2160 A$=MID(R$,P%,1%)                              ! RETRIEVE 1ST ASCII CHARACTER
2170 B$=MID(R$,P%+1%,1%)                           ! RETRIEVE 2ND ASCII CHARACTER
2180 A%=ASCII(A$)                                  ! CONVERT TO ASCII NUMBER
2190 B%=ASCII(B$)*((A%>=128%)-(A%<128%))           ! CONVERT TO ASCII NUMBER
2200 IF A%>=128% THEN A%=-(A%-128%)                ! NEGATIVE NO. HANDLER
2210 V%=256%*A%+B%                                 ! ORIGINAL NUMBER, BASE 10
2220 PRINT V%                                      ! VERIFY ORIGINAL NUMBER
2230 GOTO 2000
2240 END
```



NOTE: % signs left out for clarity

FULL TO INTEGER



NOTE: $0 \leq A \leq 127$ for positive No.s;
$128 \leq A = A + 128 \leq 255$ for negative No.s

3.16 GENERAL I/O ROM STATEMENT EQUIVALENTS

The following table gives BASIC equivalents for the 9825A General I/O ROM:

Table 3-10

General I/O ROM Statement Equivalents

9825A HPL	1720A BASIC
fmt(format) fmt 1,f8.2 wrt(write) wrt 6.1,A	PRINT #6,USING "#####.##",A
fmt 1,e8.1 wrt 6.1,A	PRINT #6, USING '###.#^^^^",A always use exactly 4 hats
fmt 1,C10,f5.1 wrt 6.1,"FUNCTION=",A	PRINT #6,"FUNCTION="; PRINT #6, USING "###.#",A
fmt 1,4f5.1 wrt 6.1,A,B,C,D	Print #6, USING "###.#",A,B,C,D
red(read) red 6, A$,X,Y	INPUT #6,A$,X,Y
wtb(write binary) wtb 6,31,"B",5	WBYTE A%(0..5)
list # list #6	SAVE "KB2:" <CR> or KB2:=/L <CR> in File Utility Program
conv (convert characters)	not needed
rdb(read binary) rdb (4)	RBYTE A% (3..5)
rds(read status) rds (6)	Use the ON ERROR GO TO statement or IEEE commands
wtc(write control)	Use IEEE commands

Program 15 - Basic Version

Trisonometric Table

DEGREES (Radians)	Sine	Cosine	Tangent
0 (0.00)	0.000	1.000	0.000E+00
10 (0.17)	0.174	0.985	1.763E-01
20 (0.35)	0.342	0.940	3.640E-01
30 (0.52)	0.500	0.866	5.774E-01
40 (0.70)	0.643	0.766	8.391E-01
50 (0.87)	0.766	0.643	1.192E+00
60 (1.05)	0.866	0.500	1.732E+00
70 (1.22)	0.940	0.342	2.747E+00
80 (1.40)	0.985	0.174	5.671E+00
90 (1.57)	1.000	0.000	INFINITY
100 (1.75)	0.985	-0.174	-5.671E+00
110 (1.92)	0.940	-0.342	-2.747E+00
120 (2.09)	0.866	-0.500	-1.732E+00
130 (2.27)	0.766	-0.643	-1.192E+00
140 (2.44)	0.643	-0.766	-8.391E-01
150 (2.62)	0.500	-0.866	-5.774E-01
160 (2.79)	0.342	-0.940	-3.640E-01
170 (2.97)	0.174	-0.985	-1.763E-01
180 (3.14)	0.000	-1.000	0.000E+00

```
1 ! 'TRIG2',          TRIG. TABLE
7 ! VERSION 1.0, 20 NOVEMBER 1979
10 !
11 ! THIS PROGRAM DEMONSTRATES THE USE OF THE "PRINT USING" STATEMENT AS
12 ! COMPARED TO THE HPL "fmt" STATEMENT
13 !
18 CLOSE 1                    ! MAKE SURE FILE HASN'T BEEN PREVIOUSLY OPENED
19 ON ERROR GOTO 145          ! CATCHES MATH ERROR FOR TAN(90DEG)
20 OPEN"KB1:"AS NEW FILE 1  ! ASSIGN FILE 1 TO KB1 PRINTER
30 PRINT #1, SPACE$(30%),"Trisonometric Table" \ PRINT #1 \ PRINT #1
40 A%=0% \ X=0 ! INITIALIZE
45 C=180/PI     ! CONVERSION FACTOR :  FROM DEGREES TO RADIANS
50 PRINT#1,"      DEGREES (Radians)";SPACE$(14%);"Sine";SPACE$(10%);"Cosine";
60 PRINT#1, SPACE$(13%);"Tangent" \ PRINT #1
70 FOR X=0 TO 180 STEP 10
80 PRINT #1, USING "S###", SPACE$(8%);X;
90 PRINT #1, USING "#.##","  (";X/C;")";
100 PRINT#1, USING "S##.###",SPACE$(16%);SIN(X/C);
110 PRINT#1, USING "S##.###",SPACE$(8%);COS(X/C);SPACE$(11%);
125 PRINT#1, USING "S#.###^^^^",TAN(X/C)
130 A%=A%+1% \ IF A%=3% THEN PRINT#1 \ A%=0%  ! SPACE AFTER EACH 3 LINES
140 NEXT X
141 CLOSE 1
142 END
145 PRINT #1, "INFINITY" \ RESUME 130
150 END
```

Program 15 - HPL

```
0: fmt 1,5/;wrt
 6,1
1: fmt 30x,"Trig
 onometric Table
 ",2/;wrt 6
2: deg;sfg 14;
 0→A→X
3: fmt 2,6x,"Deg
 rees (Radians)"
 ,12x,"Sine",
 10x,"Cosine",
 12x,"Tangent",/
4: wrt 6,2
5: fmt 3,9x,f3.0
 ," (",f4.2,")",
 15x,f6.3,9x,
 f6.3,10x,e10.3
6: wrt 6.3,X,πX/
 180,sin(X),cos(
 X),tan(X)
7: A+1→A
8: if A=3;0→A;
 fmt ;wrt 6
9: if 180>X;X+
 10→X;gto 6
10: wrt 6,1
11: end
```

Trigonometric Table

Degrees (Radians)	Sine	Cosine	Tangent
0 (0.00)	0.000	1.000	0.000E 00
10 (0.17)	0.174	0.985	1.763E-01
20 (0.35)	0.342	0.940	3.640E-01
30 (0.52)	0.500	0.866	5.774E-01
40 (0.70)	0.643	0.766	8.391E-01
50 (0.87)	0.766	0.643	1.192E 00
60 (1.05)	0.866	0.500	1.732E 00
70 (1.22)	0.940	0.342	2.747E 00
80 (1.40)	0.985	0.174	5.671E 00
90 (1.57)	1.000	0.000	9.999E 99
100 (1.75)	0.985	-0.174	-5.671E 00
110 (1.92)	0.940	-0.342	-2.747E 00
120 (2.09)	0.866	-0.500	-1.732E 00
130 (2.27)	0.766	-0.643	-1.192E 00
140 (2.44)	0.643	-0.766	-8.391E-01
150 (2.62)	0.500	-0.866	-5.774E-01
160 (2.79)	0.342	-0.940	-3.640E-01
170 (2.97)	0.174	-0.985	-1.763E-01
180 (3.14)	0.000	-1.000	0.000E 00

3.17 FLEXIBLE DISK ROM STATEMENT EQUIVALENTS*

The following table gives BASIC equivalents for the HPL statements associated with the 9885 Floppy Disk:

Table 3-11

9825A HPL	1720A	MODE
drive	The HPL drive No. is equivalent to the device No. in the PRINT, PRINT USING and OPEN statments. The HPL drive No. is equivalent to the file No. in an INPUT or INPUT LINE statment. The device number is related to the file No. via an OPEN statement. A device may be accessed by more than one file No. Refer to Appendix, 5B for a discussion of devices, channels and files.	BASIC
cat(catalog)	/L or /E ! Lists the directory of the disk	FUP
save**	SAVE <"file name"> or SAVE L <"file name">	BASIC Immediate
get***	OLD <"file name">	
chain***	RUN <"file name"> The COM (Common) statement must be used to retain common variables	N/A
resave	This command is not necessary. The BASIC Interpreter checks to see if a file already exists on the specified device; if it does, the file is replaced only after the programmer acknowledges the desireability of doing so.	N/A
Save K (Save Keys)	Not necessary. CRT attributes can be part of the basic program.	N/A
get K (get keys)	Not necessary. See save K(above).	N/A
open	OPEN <"Device:No."> AS NEW FILE <file No.>SIZE <No. of blocks>. Refer to discussion of devices, channels and files in the Appendix 5B.	BASIC

9825A HPL	1720A	MODE
kill	KILL <"file name.extension">	BASIC
	<file name>/D ! Delete file	FUP
files	OPEN<"Device:No.">AS OLD FILE <file No.> There is no return variable. See Appendix 5B for discussion of devices, channels and files. CLOSE initializes data pointers.	BASIC
asgn(assign)	not needed:strings are allowed in OPEN statements	BASIC
sprt(serial print)	PRINT #<file No.>,<list> or PRINT #<file No.>, USING <string><list> \format	BASIC**,*
sread (serial read)	INPUT #<file No.>, <list>	BASIC**,*
rprt (random print) & rread (random read)	The random access feature is available with virtual arrays. In order to OPEN a virtual array a DIM statement must be included. See Program 18 & 19 in this section. See note on Table 2-1.****	BASIC
type	Not available. All data can be input as a BASIC string or viewed as a file in the File Utility Program (FUP).	N/A
rnam(re-name)	<file name>=<file name>/R<CR> / new name \ old name	FUP
on end savem(save memory) getm(get memory)	Not available	N/A

copy	/W	FUP
dump load	Not available	N/A
repk(repack)	<device name No.:>/P ! Pack the disk	FUP
von(verify on) voff(verify off)	Verificationof disk read and write is automatic	N/A

****Legal values for dimensioning the maximum string length in Virtual arrays are powers of 2 between 2 and 512 inclusive.

*** Fluke BASIC does not support the recording of sections of programs. This can be accomplished, however, by first deleting unwanted lines (DELETE<line No.>-<line No.>) and then recording the program (SAVE 'file name').

Fluke BASIC does not support the merging of separately recorded programs into one contiguous program. Separate programs may, however, be merged by using the File Utility Program (FUP) via the Merge Utility.

```
(<file name> = <file name>, <file name> <CR>
    /                \       /         /
merged          programs to be      carriage
program         merged              return
```

* List of variables must be separated by commas

** See Appendix, 5B for discussion of device No., file No. and channel No.

9825A HPL	1720A	MODE
getb(get binary)	All system interpreters can be loaded in binary via FUP.	FUP
init (initialize)	/F	FUP
boot vfyb(verify boot)	all this happens when the RESTART button is pushed	ALL MODES
killall	/Z	FUP
dtrk(dump track) ltrk(load track)	not available	
dirc (directory)	not available, any spare directory would be on another device, e.g., the Electronic Disk. A backup disk is recommended at all times.	

Program 16 - Basic

```
1 ! 'DDATA', DATA HANDLER 4
7 ' VERSION 1.0, 20 NOVEMBER 1979
10 !
11 ' THIS PROGRAM DEMONSTRATES THE USE OF THE "INPUT LINE" STATEMENT FOR
12 ' INPUTING MANY DATA ITEMS AT ONCE FROM THE DISK.  NOTE LINE 150.  BE
13 ! SURE TO LOOK AT DATA4 .DAT IN FUP
14 '
20 CLOSE 3                                        ! MAKE SURE CHANNEL IS CLOSED
30 OPEN 'MFO:DATA4.DAT'AS NEW FILE 3 SIZE 1       ! OPEN CHANNEL
40 A$="THIS IS A TEST" \ A1=PI \ A2=2*PI          ! GENERATE DATA
50 DIM B(10%)                                     ! DIMENSION ARRAY FOR MORE DATA
60 FOR I%=0% TO 10%
70 B(I%)=I%+10%                                   ! GENERATE MORE DATA
80 NEXT I%
90 PRINT #3, A$ \ PRINT #3, A1 \ PRINT #3, A2 ! STORE DATA
100 A$=SPACE$(15%) \ A1=0 \ A2=0                  ! RE-INITIALIZE
110 PRINT #3, B(0..10)
120 CLOSE 3                                       ! REPOSITION DATA POINTER
130 OPEN "MFO:DATA4.DAT" AS OLD FILE 3 SIZE 1 ! OPEN CHANNEL
140 DIM C(10%)                                    ! ALTERNATE ARRAY
150 INPUT LINE #3,B$,B1,B2,C(0..10)
160 PRINT C(0..10)
170 PRINT B$,B1,B2                                ! DISPLAY DATA
180 CLOSE 3                                       ! CLOSE CHANNEL
190 END
```

Program 17 - Basic

```
1 ! 'HDATA', DATA HANDLER 5
7 ' VERSION 1.0, 20 NOVEMBER 1979
10 '
11 ' THIS PROGRAM SHOWS AN ALTERNATE WAY OF STORING AND RETRIEVING DATA
12 ' FROM THE DISK WITHOUT USING THE INPUT LINE STATEMENT.  BE SURE TO
13 ' LOOK AT DATA5 .DAT IN FUP.
14 !
15 CLOSE 2                                         ! MAKE SURE FILE IS CLOSED
20 DIM C(10%)
30 A$="THIS IS A TEST" \ A1=PI \ A2=2*PI          ! INITIALIZE
40 OPEN "DATA5.DAT" AS NEW FILE 2 SIZE 1          ! OPEN CHANNEL
50 PRINT #2, A$;',';A1;',';A2;                    ! STORE DATA
55 PRINT #2
60 FOR I%=0% TO 10%
70 INPUT C(I%)                                    ! INPUT DATA
80 PRINT #2, C(I%);                               ! STORE DATA
90 IF I% <> 10% THEN PRINT #2, ','; ELSE PRINT #2
100 NEXT I%
110 CLOSE 2                                       ! REPOSITION DATA POINTER
120 DIM B(10)
130 OPEN "DATA5.DAT" AS OLD FILE 2 SIZE 1         ! OPEN CHANNEL
140 INPUT #2, B$, B1, B2
145 INPUT #2, B(0..10)
150 PRINT B(0..10)
160 PRINT B$, B1, B2                              ! DISPLAY DATA
170 CLOSE 2                                       ! CLOSE CHANNEL
180 END
```

Program 18 - Basic

```
1 ! 'VARAY1', DATA HANDLER 6A: VIRTUAL ARRAY
7 ! VERSION 1.0, 20 NOVEMBER 1979
10 !
100 !*****************************************************************************
110 !
120 ! VIRTUAL ARRAYS RESIDE ON THE DISK INSTEAD OF IN MAIN MEMORY.  VIRTUAL
130 ! ARRAYS ARE NOT LOST WHEN A NEW PROGRAM IS CHAINED IN.  OTHERWISE, AS
140 ! FAR AS THE FUNCTIONS OF MATHEMATICS OR ASSIGNMENT ARE CONCERNED VIRTUAL
150 ! ARRAYS ARE HANDLED AS IF THEY WERE VIRTUALLY IN THE MAIN MEMORY, EXCEPT
160 ! FOR THE FACT THAT THEY DO NOT REQUIRE ANY OF THE MAIN MEMORY FOR
170 ! STORAGE.  THE USE OF VIRTUAL ARRAYS ALLOWS THE PROGRAMMER TO USE MORE
180 ! OF THE MAIN MEMORY FOR PROGRAM RESIDENCE.
190 !
200 ! VIRTUAL ARRAYS ARE RANDOMLY STORED AND RETRIEVED FROM THE DISK.  RUN
210 ! 'VARAY1' AND 'VARAY2' IS CHAINED IN.  NEXT RUN 'VARAY2' AFTER LEAVING
220 ! BASIC TO GO TO THE CONSOLE MONITOR TO PROVE THE ARRAYS STILL EXIST.
230 !
2000 CLOSE 2     ! MAKE SURE FILE HAS NOT BEEN OPEN FOR ANOTHER PURPOSE
2010 OPEN "MF0:RESULT" AS NEW DIM FILE 2 SIZE 7  ! OPEN VA FILE ON DISK
2020 DIM#2, A(300%), C0$(10%)=64, B(5%)          ! DIMINSION VA'S
2022 !
2024 !      STORE DATA IN THE VIRTUAL ARRAYS
2026 !
2030 FOR I%=1% TO 300% STEP 4%
2040 FOR J%=0% TO 3%
2050 A(I%+J%)=2%*(I%+J%)
2060 NEXT J%
2080 NEXT I%
2100 C0$(1%)="VIRTUAL ARRAYS" \ C0$(2%)="ARE USEFUL" \ B(1%)=PI
2110 CLOSE 2
2120 RUN"VARAY2"    ! CHAIN IN NEW PROGRAM
2130 END
```

Program 19 - Basic

```
1 ! 'VARAY2', DATA HANDLER 6B: VIRTUAL ARRAY
7 ! VERSION 1.0, 20 NOVEMBER 1979
10 !
100 ! THIS PROGRAM IS CHAINED IN AFTER 'VARAY1' TO PROVE THAT CHAINING IN
110 ! A NEW PROGRAM DOES NOT DESTROY VIRTUAL ARRAYS
120 !
2000 CLOSE 3
2010 OPEN "MF0:RESULT" AS OLD DIM FILE 3
2020 DIM #3, A1(300%), C1$(10%)=64, B1(5%)
2030 FOR I% =1% TO 300% STEP 4%
2040 PRINT A1(I%), A1(I%+1%), A1(I%+2%), A1(I%+3%)
2050 NEXT I%
2060 PRINT "A(150) = ", A1(150%)
2070 PRINT C1$(1%..2)
2080 PRINT B1(1%)
2090 CLOSE 3
2100 END
```

3.18 EXTENDED I/O ROM PROGRAMMING STATEMENT EQUIVALENTS

The following table shows statement equivalents for the IEEE-488 statements:

Table 3-12

Extended I/O ROM Programming Statement Equivalents

9825A HPL	1720A BASIC
Binary Mode Statements	
moct(octal mode) mdec(decimal mode)	Not available
Bit Functions	
ior(inclusive or)	OR
eor(exclusive or)	XOR
band(binary and)	AND
cmp(compliment)	
rot(rotate)	
shf(shift)	
add	not available
bit	
dto(decimal to octal)	
otd(octal to decimal)	
HP-IB Control Statements	**IEEE Commands**
dev(device)	Use device number
cmd(command)	Not available
equ(equate)	
trg trig 7	TRIG@<device list>
clr(clear) clr 7	CLEAR@<device list>
cli(clear interface) cli 7	INIT [PORT<expr.>]
rem(remote)	REMOTE@<device>
lcl(local)	LOCAL@<device>
llo(local lockout) llo 7	LOCKOUT [PORT,expr.>]
polc(poll configure)	CONFIG@<*>TO<device>
polu(poll unconfigure)	CONFIG@<*>

* PRIORITY

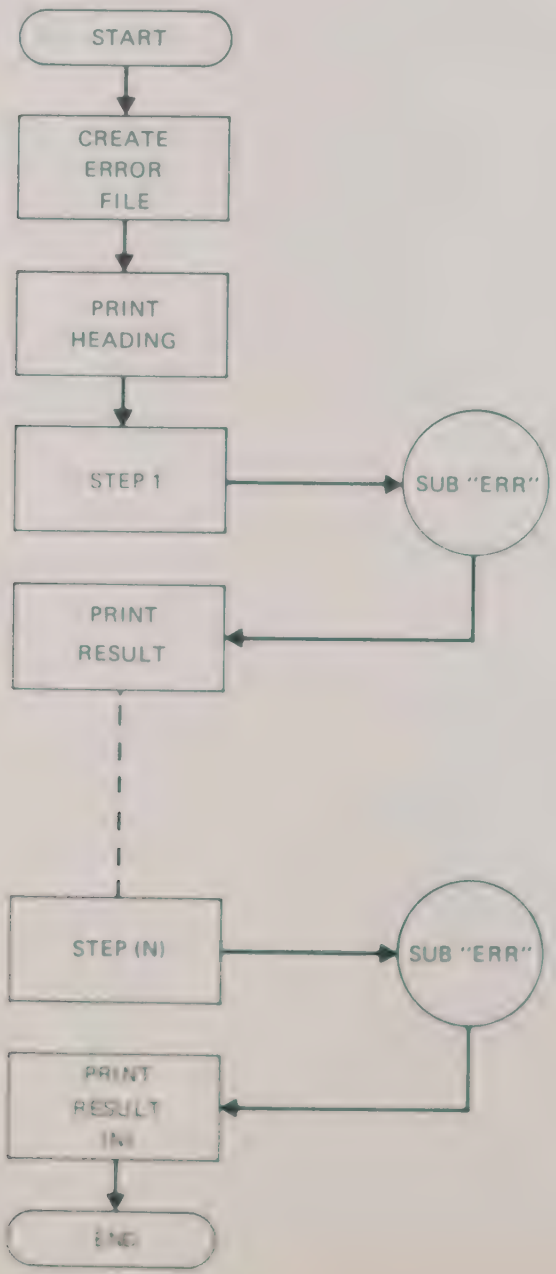
pol(parallel poll) pol 7	ON PPOL
[PORT<expr.>]GOTO	<line No.>
pct(pass control)	not available
rqs(require service)	ON SRQ
rds(extended read status)	not available
	*PRIORITY
Conversion Statements	
ctbl(conversion table)	Not Available
par(parity)	
Interrupt Statements	IEEE Commands
oni(on interrupt)oni S"<label>"	ON SRQ [PORT<expr>] <line No.>
eir(enable interrupt)eir S eir S,M	See above (ON SRQ)
iret(return from interrupt) iret	RESUME
Error Recovery	STANDARD STATEMENTS
on err on err "<label>"	ON ERROR GOTO <line NO.>
rom(read only memory)	Not necessary - 1720A has no hardware ROMs
ern(error number)	ERR
erl(error line)	ERL
Buffered I/O Statement	
buf(buffer)	I/O is always buffered
tfr(transfer)	not necessary-Operating System transfers data from buffer
Time Statement	
time time T	TIME or TIME$
Interface Contrcl Statement	
wti (write interface)	Not available
Interface Control Functions	
rdi(read interface)	
iof(I/O flag)	
ios(I/O Status)	Not available

Program 20 - HPL Version

9825A - 5100A

Application — Multimeter verification program

Following program is an example of a typical verification test program. The results are printed after the error control of the 5100A has been used. This program can easily be extended or changed, to verify other instruments.



```
================
   8 0 2 0 A
================
```

Change line 13 for other instruments

```
DATE.............
```

line 15

```
NUMBER...........
```

line 16

```
******VDC*******
```

Change 3 characters text in line 17 for other function.

```
RNG  INPUT  ERR(%)
---------------
R1   +.19V   -0.11
R2   +1.9V    0.00
R3   +19 V    0.11
R4   +190V   -0.16
R5   +500V    0.60
```

```
******VAC*******
```

line 40

```
R1    .19V   -0.11
R2    1.9V    0.00
R3    19 V   -0.74
R4    190V  *Fail*
R5    500V   -0.80
```

```
******OHM*******
```

line 61

```
R1    100Ω   -0.20
R2     1kΩ  *Fail*
R3    10kΩ    0.00
R4   100kΩ    0.10
R5     1MΩ    0.20
R6    10MΩ   -0.20
```

```
*******nS*******
```

line 91

```
R1   200nS    0.80
```

```
******DC*******
```

line 97

```
R1   1.9mA   -0.05
R2   19 mA    0.11
R3   190mA    0.00
R4    1.9A   -0.05
```

```
******AAC*******
```

line 114

```
R1   1.9mA    0.11
R2    19mA   -0.11
R3   190mA    0.11
R4    1.9A   -0.26
```

```
TESTED..........
```

line 131

- format 1, used for line 17, 40, 61, 91, 97, 114

- format 2, used for line 19

- format 3, used for the printed results

- format 4, used for the printed results and generating the omega character

T1 - T11 containing the maximum allowed errors for 11 different ranges.
In this case for the 8020A.

Example:

T1, used for VDC = 0.25% + 1 digit.
At 19 Volt, this is 0.0025 + 1/1900 = 0.003

- 5100A reset

- Printing xxxxxx VDC xxxxxxx

- Printing RNG INPUT ERR(%

Printing . . . . . .

```
20: "200mVDC"→B$
21: "+0.19V,N@<<
 "→C$
22: cll 'err'(T[
 1])
23: wrt 16.3,
 "R1","+.19V",A$
24: "2VDC"→B$
25: '1.9V,N@<<"→
 C$
26: cll 'err'(T[
 1])
27: wrt 16.3,
 "R1","+1.9V",A$
28: "20VDC"→B$
29: '19V,N@<<"→C
 $
30: cll 'err'(T[
 1])
31: wrt 16.3,
 "R3","+19 V",A$
32: '200VDC"→B$
33: '190V,N@<<"→
 C$
34: cll 'err'(T[
 1])
35: wrt 16.3,
 "R1","+190V",A$
36: '200VDC"→B$
37: '19V,N@<<"→
 C$
38: cll 'err'(T[
 1])
39: wrt 16.3,
 "R5","+500V",A$
40: spl 1;wrt
 1[...],"VAC";spl
 1
41: '200mVAC"→B$
42: "0.19V400-,
 N@<<"→C$
43: cll 'err'(T[
 3])
44: wrt 16.3,
 "R1","+.19V",A$
45: '"VAC"→B$
46: '1.9V,N@<<"→
 C$
47: cll 'err'(T[
 2])
48: wrt 16.3,
 "R2","+1.9V",A$
49: '"VAC"→B$
50: '19V,N@<<"→C
 $
51: cll 'err'(T[
 2])
52: wrt 16.3,
 "R3","+19 V",A$
```

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6

Step 7

Step 8

B$ is part of the text on the display. C$ is the command string for the 5100A. The program first jumps to the sub routine, called "err", starting at line 133, and using T1 error condition. Line 23 prints the result after the subroutine has been completed. "R1" is text that can be changed. "+.19V" is text. A$ is the calculated error, returned by the 5100A in the subroutine.

and so on up to line 131

```
53: "200VAC"→B$
54: "190V,N@<<"→
 C$
55: cll 'err'(T[
 3])
56: wrt 16.3,
 "R4","190V",A$
57: "750VAC"→B$
58: "500V,N@<<"→
 C$
59: cll 'err'(T[
 4])
60: wrt 16.3,
 "R4","500V",A$
61: spc 1;wrt
 16.1,"AAC";spc
 1
62: dsp "NEXT
 STEP COMPENSATE
 S THE LEADS. "
 stp
63: "200"&char(1
 7)→B$
64: "1Z,N@<<<<"→
 C$:1→F
65: cll 'err'(1)
66:
67: "100Z,N@<<"→
 C$
68: cll 'err'(T[
 5])
69: wrt 16.4,
 "R1","100",17,
 A$
70: "2"&char(17
 )→B$
71: "1000Z,N@
 →C$
72: dsp "----(1)
 "
73:
74: wrt 16.4,
 "R2","1k",17,A$
75: "10"&char(1
 7)→B$
76: "1E-7,N@
 →C$
77:
78:
79:
80:
81: char
 )→B$
82: "1E+6Z,N@<<"
 →C$
83: cll 'err'(T[
 6])
84: wrt 16.4,
 "R5","1M",17,A$
85: "20M"&char(1
 7)→B$
86: "1E+7Z,N@<<"
 →C$
87: cll 'err'(T[
 7])
88: wrt 16.4,
 "R6","10M",17,
 A$
89: "200nS"→B$
90: spc 1;wrt
 16.1,"+nS";spc
 1
91: "1E+7Z,N@
 →C$
92: cll 'err'(T[
 8])
93: wrt 16.3,
 "R1","200nS",A$
94: dsp "CONNECT
 CABLE TO THE..
 ....(cont)";
 stp
95: dsp "CURRENT
 INPUT+HI/LO....
 ...(cont)";
 stp
96: spc 1;wrt
 16.1,"ADC";spc
 1
97: "2mADC"→B$
98: "+1.9E-3A,
 N@"→C$
99: cll 'err'(T
 [9])
100: wrt 16.3,
 "R1","1.9mA",A$
101: "20mADC"→B$
102: "19E-3A,
 N@<<"→C$
103: cll 'err'(T
 [9])
104: wrt 16.3,
 "R2","19mA",A$
105: "200mADC"→B
 $
106: "+190E-3A,
 N@<<"→C$
107: cll 'err'(T
 [9])
108: wrt 16.3,
 "R3","190mA",A$
109: "2000mADC"→
 B$
110: "1.9A,N@<<"
 →C$
111: cll 'err'(T
 [9])
112: wrt 16.3,
 "R4","1.9A",A$
113: spc 1;wrt
 16.1,"AAC";spc
 1
114: "2mAAC"→B$
115: "1.9E-3A400
 H,N@<<"→C$
116: cll 'err'(T
 [10])
117: wrt 16.3,
 "R1","1.9mA",A$
118: "20mAAC"→B$
119: "19E-3A,
 N@<<"→C$
120: cll 'err'(T
 [11])
121: wrt 16.3,
 "R2","19mA",A$
122: "200mAAC"→B
 $
123: "0.19A,N@<<
 "→C$
124: cll 'err'(T
 [11])
125: wrt 16.3,
 "R3","190mA",A$
126: "2000mAAC"→
 B$
127: "1.9A,N@
 →C$
128: cll 'err'(T
 [11])
129: wrt 16.3,
 "R4","1.9A",A$:
 spc 1
130: prt "TESTED
 ..........";
 end
```

"err" subroutine

```
103: "err":
104: dsp "SELECT
   '&B$&" RANGE.(
 cont)";stp
105: wrt C,C$
106: dsp "USE
 5100A ERROR
 CONTROL.  (cont
 )";wait 1000
107: lcl C;stp
108: if X=1;wtb
 C,"$";jmp 2
109: wtb C,"?";
 red C,X$;val(X$
   )
110: if abs(A/
 100)>p1;"*Fail*
  "&A$;jmp 2
111: str(A)→A$
112: wrt C,"S"
113: end
```

- Displays: Select (text of B$) Range. (cont).
- Programs the 5100A
- Displays: Use 5100A error control. (cont).
- Waits 1 second to allow the 5100A to settle.
- Places the 5100A into local operation, to allow the operator to use the error control.
- Asks for the central display, containing the 8020A error.
- Compares the error to the allowable error, Tn which is p1.
  If the error exeeds the T limit, the text "x fail x" comes in A$ instead of the error.
  Line 142 places the 5100A into standby.

Line 138 programs the new ohms reference, only if X = 1. X is set to 1 in line 64 and is reset in line 66.

5100 - 8020 Verification Program

Purpose: It demonstrates how the previous 9825A program can be made less complicated, and even more safe, by the use of the 1720A.

Description: This program is a verification test for the 8020A DMM. It could be modified to be used with any other Multimeter. The program uses the CRT of the 1720A to set and edit on the 5100A to the correct range and instructs the operator to make necessary adjustments on the 8020A to accomplish the test. Through the use of the editing functions (Note, operated from the CRT of the 1720A) % of error on the DMM can be determined. If DMM is out of specs the program will generate a failure prompt for that range. Appropriate safety prompts are displayed, and the 5100A is always on standby, until operator is ready to make test. After all ranges have been checked, program ends. Program also has error detection for the IEEE bus, and returns suggestions for the correction of the problem.

5100A address is set at 3

CAUTION - THE 5100A CAN HAVE AN OUPUT AS HIGH AS 750 VOLTS

Program 20

Basic Version

```
1!   5100 STAND ALONE PROGRAM
2!
3!
4!
5!   DATE 20-10-79    12:18:25
6!
10!      THIS PROGRAM IS AN 8020 VERIFICATION ROUTINE.  IT WAS TRANSLATED FROM
20!  THE HPL PROGRAM WRITTEN BY JAN GOVERTS.  THE PROGRAM HAS THE APPROPRIATE
30!  CHANGES IN IT, TO MAKE IT MORE USEFUL ON THE 1720A.
31!
32!                          THE CRT KEYBOARD
33!
34!     ------------------------------------------------------------
35!    |                                                            |
36!    |  [1 | 2]   [4 | 5]                                         |
37!    |                                                            |
38!    |  [21122]   [24125]                                         |
39!    |                                                            |
40!    |                                                            |
41!    |                                      [56157]   [59160]     |
42!    |                                                            |
43!     ------------------------------------------------------------
49!
50!           THE FOLLOWING ARE THE VARIBLES FOR THE DISPLAY GRAPHICS
60!
100 ES$=CHR$(27%)+'['     !ESCAPE SEQUENCE
125 CH$=ES$+'2J'+ES$+';H'!CLEAR AND HOME
150 BP$=CHR$(7%)+CHR$(7%)+CHR$(7%)+CHR$(7%)!THE BEEPER
175 NS$=ES$+'p'           !NORMAL SIZE CHARACTERS
200 DS$=ES$+'1p'          !DOUBLE SIZE CHARACTERS
225 HL$=ES$+'1m'          !HIGH LIGHT
250 IV$=ES$+'7m'          !INVERSE VIDEO
275 AO$=ES$+'m'           !ATTRIBUTES OFF
300 BG$=ES$+'5m'          !BLINKING
325 CS$=ES$+'2K'          !CLEAR LINE
326 BL$=ES$+'16;H'        !CURSOR TO BOTTOM LEFT
350 X=0                   !THIS IS THE VARIBLE FOR SETTING THE REFERENCE
351!
352 CL%=3%                !THIS IS THE VARIBLE FOR SETTING OF THE CALIBRATOR
375!
400 DIM Y%(60),Y0%(60),Y1%(60),Y2%(60),Y3%(60),N$(14),T(10)
425!
426!           THE FOLLOWING IS WHAT SETS UP OUR DECISION ARRAYS
427!
450  DATA 1,2,4,5,21,22,24,25,39,40,56,57,59,60,0
475  READ A%                                        !THIS IS THE ARRAY FOR THE
500  FOR I%=0% TO 60%                               !VALID KEYS.
525  IF I%=A% THEN Y%(I%)=2%\READ A% ELSE Y%(I%)=1%
550  NEXT I%
575!
600 DATA 1,4,0
625 READ A%                                         !THIS IS THE ARRAY FOR
650 I=2                                             !VALID KEYS UP AND DN.
675 FOR I%=0% TO 60%
700 IF I%=A% THEN Y0%(I%)=1\I%=I%+1%\Y0%(I%)=1\I=I+1\READ A% ELSE Y0%(I%)=1%
725 NEXT I%
750'
775 DATA 39,56,0
800 READ A%                                         !THIS IS THE ARRAY FOR
825 I=2                                             !THE GO KEY.
```

```
850 FOR IZ=0Z TO 60Z
875 IF IZ=AZ THEN Y1Z(IZ)=I\IZ=IZ+1Z\Y1Z(IZ)=I\I=I+1\READ AZ ELSE Y1Z(IZ)=1Z
900 NEXT IZ
925'
950 DATA 21,24,0
975 READ AZ                                          !THIS IS THE VALID KEYS
1000 I=2                                             !FOR THE <,> KEYS.
1025 FOR IZ=0Z TO 60Z
1050 IF IZ=AZ THEN Y2Z(IZ)=I\IZ=IZ+1Z\Y2Z(IZ)=I\I=I+1\READ AZ ELSE Y2Z(IZ)=1Z
1075 NEXT IZ
1100'
1125 DATA 59,0
1150 READ AZ                                         !THIS IS THE ARRAY FOR
1175 I=2                                             !THE NEXT KEY
1200 FOR IZ=0Z TO 60Z
1225 IF IZ=AZ THEN Y3Z(IZ)=I\IZ=IZ+1Z\Y3Z(IZ)=I\I=I+1\READ AZ ELSE Y3Z(IZ)=1Z
1250 NEXT IZ
1275'
1276!            THIS IS THE DATA FOR OUR LIMITS FOR THE 8020
1277!
1300 DATA .003,.0065,.0086,.014,.006,.003,.021,.03,.008,.021,.016
1325 READ T(0..10)
1326!
1350 PRINT CH$;DS$;                                  !AND WE CLEAR,AND DS SCREEN
1351'
1352!            AND WE PRINT THE DISPLAY HERE
1353'
1375 PRINT CPOS(2,12);A0$;CPOS(2,5);IV$;CPOS(2,21);A0$;CPOS(2,14);IV$;
1400 PRINT CPOS(7,27);A0$;CPOS(7,20);IV$;CPOS(4,12);A0$;CPOS(4,5);IV$;
1425 PRINT CPOS(4,21);A0$;CPOS(4,14);IV$;CPOS(7,36);A0$;CPOS(7,29);IV$;
1450 PRINT CPOS(2,8);'UP';CPOS(2,17);'DN';CPOS(7,22);'NEXT';CPOS(4,8);'<';
1475 PRINTCPOS(4,18);'>';CPOS(7,32);'GO';CPOS(5,36);A0$;CPOS(5,29);IV$;' EXIT';
1500 B$='SELECT '\B1$=' RANGE'
1501'
1502!            THIS IS THE CAUTIONARY PROMPT FOR HIGH VOLTAGE
1503'
1525 SS$= CPOS(7,7)+'CAUTION!'+CPOS(7,15)+A0$+CPOS(7,4)+HL$+EG$
1550 ST$= CPOS(8,5)+'HIGH VOLTAGE'+CPOS(8,17)+A0$+CPOS(8,3)+HL$+EG$
1551'
1552!            AND THE OTHER HOUSE KEEPING NEEDS
1553'
1575 S0$= CPOS(6,1)+C5$
1600 S1$=CPOS(7,6)+'          '
1625 S2$=CPOS(8,4)+'             '
1626 S3$=CPOS(2,25)+'   CHANGE     '
1627 S4$=CPOS(3,25)+'  RANGE NOW   '
1628 S5$= CPOS(2,25)+'               '+CPOS(3,25)+'               '
1629'
1630 ON CTRL/C GOTO 32260
1631 ON ERROR GOTO 32030
1632'
1675 INIT PORT 0       'AND NOW WE INIT THE 5100
1700 CLEAR @CLZ        'AND BUSS
1701'
1724 PRINT S0$;S3$;S4$;
1725 PRINT CPOS(6,6);B$;'200mVDC';B1$;     'THIS IS THE 200mV RANGE VER
1730 KYZ=KEY\ON YJZ(KYZ) GOTO 1730,1750   !PORTION
1750 C$='+0.19V.##'
1775 DATA ' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
1800 READ N$(0..13)
```

```
1825 PX=13%
1850 AX=0%
1875 GOSUB 8525
1876 PRINT S3$;S4$;
1877!
1900 PRINT S0$;                              !AND THIS IS THE 2V RANGE
1925 PRINT CPOS(6,6);B$;'2VDC';B1$;          !FROM NOW ON IT SHOULD BE OBVIOUS
1950 KY%=KEY\ON Y3%(KY%) GOTO 1950,1975
1975 C$='1.9V,N@'
2000 DATA' ','1',' ','.',' ','9',' ','0',' ','0',' ','0',' ','0'
2025 READ N$(0..13)
2050 PX=13%
2075 AX=0%
2100 GOSUB 8525
2101 PRINT S3$;S4$;
2102!
2125 PRINT S0$;
2150 PRINT CPOS(6,6);B$;'20VDC';B1$;
2175 PRINT SS$;ST$;
2200 KY%=KEY \ ON Y3%(KY%) GOTO 2200,2225
2225 C$='19V,N@'
2250 DATA' ','1',' ','9',' ','.',' ','0',' ','0',' ','0',' ','0'
2275 READ N$(0..13)
2300 PX=13%
2325 AX=0%
2350 GOSUB 8525
2351 PRINT S3$;S4$;
2352!
2375 PRINT S0$;
2400 PRINT CPOS(6,6);B$;'200VDC';B1$;
2425 PRINT SS$;ST$;
2450 KY%=KEY\ON Y3%(KY%) GOTO 2450,2475
2475 C$='190V,N@'
2500 DATA' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
2525 READ N$(0..13)
2550 PX=13%
2575 AX=0%
2577!
2600 GOSUB 8525
2601 PRINT S3$;S4$;
2602!
2625 PRINT S0$;
2650 PRINT CPOS(6,6);B$;'2000VDC';B1$;
2675 PRINT SS$;ST$;
2700 KY%=KEY\ON Y3%(KY%) GOTO 2700,2725
2725 C$='500V,N@'
2750 DATA' ','5',' ','0',' ','0',' ','.',' ','0',' ','0'
2775 READ N$(0..11)
2800 PX=11%
2825 AX=1%
2850 GOSUB 8525
2851 PRINT S3$;S4$;
2852!
2875 PRINT S0$;
2900 PRINT CPOS(6,6);B$;'200mVAC';B1$;
2925 PRINT S1$;S2$;
2950 KY%=KEY\ON Y3%(KY%) GOTO 2950,2975
2975 C$='0.19V400H,N@'
3000 DATA' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
3025 READ N$(0..13)
```

```
3050 PZ=13Z
3075 AZ=2Z
3100 GOSUB 8525
3101 PRINT SJ$;S4$;
3102'
3125 PRINT SO$;
3150 PRINT CPOS(6,6);B$;'2VAC';B1$;
3175 KYZ=KEY\ON Y3Z(KYZ) GOTO 3175,3200
3200 C$='1.9V,M@'
3225 DATA' ','1',' ','.',' ','9',' ','0',' ','0',' ','0',' ','0'
3250 READ N$(0..13)
3275 PZ=13Z
3300 AZ=2Z
3325 GOSUB 8525
3326 PRINT SJ$;S4$;
3327'
3350 PRINT SO$;
3375 PRINT CPOS(6,6);B$;'20VAC';B1$;
3400 PRINT SS$;ST$;
3425 KYZ=KEY\ON Y3Z(KYZ) GOTO 3425,3450
3450 C$='19V,M@'
3475 DATA' ','1',' ','9',' ','.',' ','0',' ','0',' ','0',' ','0'
3500 READ N$(0..13)
3525 PZ=13Z
3550 AZ=2Z
3575 GOSUB 8525
3576 PRINT SJ$;S4$;
3600 PRINT SO$;
3625 PRINT CPOS(6,6);B$;'200VAC';B1$;
3650 KYZ=KEY\ON Y3Z(KYZ) GOTO 3650,3675
3675 C$='190V,M@'
3700 DATA' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
3725 READ N$(0..13)
3750 PZ=13Z
3775 AZ=2Z
3800 GOSUB 8525
3801 PRINT SJ$;S4$;
3802'
3825 PRINT SO$;
3850 PRINT CPOS(6,6);B$;'750VAC';B1$;
3875 KYZ=KEY\ON Y3Z(KYZ) GOTO 3875,3900
3900 C$='500V,M@'
3925 DATA' ','5',' ','0',' ','0',' ','.',' ','0',' ','0'
3950 READ N$(0..11)
3975 PZ=11Z
4000 AZ=3Z
4025 GOSUB 8525
4026 PRINT SJ$;S4$;
4027'
4050 PRINT SO$;
4075 PRINT S1$;S2$;
4100 PRINT CPOS(6,6);'NEXT STEP COMPENSATES THE LEADS': 'NOTE HERE COMPENSATION
4125 WAIT 2000                                          'FOR LEAD RESISTANCE IS
4150 PRINT SO$;                                         'DONE.
4175 PRINT CPOS(6,6);B$;'200';CHR$(242);B1$;            'IN THE 200 OHM RANGE
4200 X=1                                                'AND HERE WE TELL THE
4225 KYZ=KEY\ON Y3Z(KYZ) GOTO 4225,4250                 'POKE'ME TO SET THE REF
4250 C$='12,M@'                                         'NOTE THAT IS VARIBLE X
4275 DATA' ','1',' ','.',' ','0',' ','0',' ','0',' ','0',' ','0'
4300 READ N$(0..13)
```

```
4325 PZ=13Z
4350 GOSUB 8525
4351 PRINT S5$;
4353!
4375 PRINT CPOS(2,25);'  LEADS NOW  ';
4400 PRINT CPOS(3,25);' COMPENSATED ';
4401 WAIT 2000
4425 C$='100Z,N@'
4450 DATA' ','1',' ','0',' ','0',' ','.',' ','0',' ','0',' ','0'
4475 READ N$(0..13)
4500 PZ=13Z
4525 AZ=4Z
4550 GOSUB 8525
4551 PRINT S3$;S4$;
4552!
4575 PRINT S0$;
4600 PRINT CPOS(6,6);B$;'2K';CHR$(24Z);B1$;
4625 KYZ=KEY\ON YJZ(KYZ) GOTO 4625,4650
4650 X=0
4675 C$='1000Z,N@'
4700 DATA' ','1',' ','0',' ','0',' ','0',' ','.',' ','0',' ','0'
4725 READ N$(0..13)
4750 PZ=13Z
4775 AZ=5Z
4800 GOSUB 8525
4801 PRINT S4$;S3$;
4802!
4825 PRINT S0$;
4850 PRINT CPOS(6,6);B$;'20K';CHR$(24Z);B1$;
4875 KYZ=KEY\ON YJZ(KYZ) GOTO 4875,4900
4900 C$='1E+4Z,N@'
4925 DATA' ','1',' ','0',' ','.',' ','0',' ','0',' ','0',' ','0'
4950 READ N$(0..13)
4975 PZ=13Z
5000 AZ=5Z
5025 GOSUB 8525
5026 PRINT S3$;S4$;
5027!
5050 PRINT S0$;
5075 PRINT CPOS(6,6);B$;'200K';CHR$(24Z);B1$;
5100 KYZ=KEY\ON YJZ(KYZ) GOTO 5100,5125
5125 C$='1E+5Z,N@'
5150 DATA' ','1',' ','0',' ','0',' ','.',' ','0',' ','0',' ','0'
5175 READ N$(0..13)
5200 PZ=13Z
5225 AZ=5Z
5250 GOSUB 8525
5251 PRINT S3$;S4$;
5252!
5275 PRINT S0$;
5300 PRINT CPOS(6,6);B$;'2M';CHR$(24Z);B1$;
5325 KYZ=KEY\ON YJZ(KYZ) GOTO 5325,5350
5350 C$='1E+6Z,N@'
5375 DATA' ','1',' ','.',' ','0',' ','0',' ','0',' ','0',' ','0'
5400 READ N$(0..13)
5425 PZ=13Z
5450 AZ=5Z
5475 GOSUB 8525
5476 PRINT S3$;S4$;
5477!
```

```
5500 PRINT SO$;
5525 PRINT CPOS(6,6);B$;'20M';CHR$(24%);B1$;
5550 KY%=KEY\ON Y3%(KY%) GOTO 5550,5575
5575 C$='1E+7Z,M@'
5600 DATA' ','1',' ','0',' ','.',' ','0',' ','0',' ','0',' ','0'
5625 READ N$(0..13)
5650 P%=13%
5675 A%=6%
5700 GOSUB 8525
5701 PRINT S3$;S4$;
5702'
5725 PRINT SO$;
5750 PRINT CPOS(6,6);B$;'200nS';B1$;
5775 KY%=KEY\ON Y3%(KY%) GOTO 5775,5800
5800 C$='1E+7Z,M@'
5825 DATA' ','1',' ','0',' ','.',' ','0',' ','0',' ','0',' ','0'
5850 READ N$(0..13)
5875 P%=13%
5900 A%=7%
5925 GOSUB 8525
5926 PRINT S5$;
5927'
5950 PRINT SO$;                                          !AND HERE WE TELL THE
5975 PRINT CPOS(6,6);'CONNECT CABLE TO THE....(cont)';   'OPERATOR TO DO THE
6000 WAIT 2000\PRINT SO$;                                !THE HOOKUPS FOR
6025 PRINT CPOS(6,6);'CURRENT INPUT, AND....(cont)';     !CURRENT MEASUREMENT
6050 WAIT 2000\PRINT SO$;
6075 PRINT CPOS(6,6);B$;'2mADC';B1$;
6076 PRINT S3$;S4$;
6077 PRINT @CL%,'!?'
6078 INPUT LINE @CL%,Z$
6079 IF RIGHT(Z$,1%)='5' THEN PRINT SO$;CPOS(6,6);'PLEASE CONNECT FOR CURRENT';
6080 IF RIGHT(Z$,1%)='5' THEN WAIT 4000 \ GOTO5950
6100 KY%=KEY\ON Y3%(KY%) GOTO 6100,6125
6125 C$='1E-3A,M@'
6150 DATA' ','1',' ','.',' ','0',' ','0',' ','0',' ','0',' ','0'
6175 READ N$(0..13)
6200 P%=13%
6225 A%=8%
6250 GOSUB 8525
6251 PRINT S3$;S4$;
6252'
6275 PRINT SO$;
6300 PRINT CPOS(6,6);B$;'20mADC';B1$;
6325 KY%=KEY\ON Y3%(KY%) GOTO 6325,6350
6350 C$='1E-3A,M@'
6375 DATA' ','1',' ','9',' ','.',' ','0',' ','0',' ','0',' ','0'
6400 READ N$(0..13)
6425 P%=13%
6450 A%=8%
6475 GOSUB 8525
6476 PRINT S3$;S4$;
6477'
6500 PRINT SO$;
6525 PRINT CPOS(6,6);B$;'200mADC';B1$;
6550 KY%=KEY\ON Y3%(KY%) GOTO 6550,6575
6575 C$='.19A,M@'
6600 DATA' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
6625 READ N$(0..13)
6650 P%=13%
```

```
6675 AZ=8%
6700 GOSUB 8525
6701 PRINT SJ$;S4$;
6702'
6725 PRINT SO$;
6750 PRINT CPOS(6,6);B$;'2000■ADC';B1$;
6775 KY%=KEY\ON Y3%(KY%) GOTO 6775,6800
6800 C$='1.9A,N@'
6825 DATA' ','1',' ','.',' ','9',' ','0',' ','0',' ','0',' ','0'
6850 READ N$(0..13)
6875 P%=13%
6900 A%=8%
6925 GOSUB 8525
6926 PRINT SJ$;S4$;
6927'
6950 PRINT SO$;
6975 PRINT CPOS(6,6);B$;'2■AAC';B1$;
7000 KY%=KEY\ON Y3%(KY%) GOTO 7000,7025
7025 C$='1.9E-3A400H,N@'
7050 DATA' ','1',' ','.',' ','9',' ','0',' ','0',' ','0',' ','0'
7075 READ N$(0..13)
7100 P%=13%
7125 A%=9%
7150 GOSUB 8525
7151 PRINT SJ$;S4$;
7152'
7175 PRINT SO$;
7200 PRINT CPOS(6,6);B$;'20■AAC';B1$;
7201 KY%=KEY\ON Y3%(KY%) GOTO 7201,7225
7225 C$='19E-3A,N@'
7250 DATA' ','1',' ','9',' ','.',' ','0',' ','0',' ','0',' ','0'
7275 READ N$(0..13)
7300 P%=13%
7325 A%=10%
7350 GOSUB 8525
7351 PRINT SJ$;S4$;
7352'
7375 PRINT SO$;
7400 PRINT CPOS(6,6);B$;'200■AAC';B1$;
7425 KY%=KEY\ON Y3%(KY%) GOTO 7425,7450
7450 C$='.19A,N@'
7475 DATA' ','1',' ','9',' ','0',' ','.',' ','0',' ','0',' ','0'
7500 READ N$(0..13)
7525 P%=13%
7550 A%=10%
7575 GOSUB 8525
7576 PRINT SJ$;S4$;
7577'
7600 PRINT SO$;
7625 PRINT CPOS(6,6);B$;'2000■AAC';B1$;
7650 KY%=KEY\ON Y3%(KY%) GOTO 7650,7675
7675 C$='1.9A,N@'
7700 DATA' ','1',' ','.',' ','9',' ','0',' ','0',' ','0',' ','0'
7725 READ N$(0..13)
7750 P%=13%
7775 A%=10%
7800 GOSUB 8525
7801 PRINT SJ$;S4$;
7802'
7825 PRINT SO$;                    !AND NOW WE CLEAN THINGS UP AND TELL THE OP
```

```
7850 PRINT CH$;                 !THAT THE 8020 HAS BEEN TESTED.
7875 PRINT CPOS(4,15);'8020 TESTED';
7880 PRINT @CLZ,'x';
7900 WAIT 8000
7925 RUN'DEMO'
7950'
7975 PRINT CPOS(3,26);'     FAIL      ';!THIS SUBR. CHECKS INPUT TO SEE IF IT
8000 FOR IZ=0 TO 5Z                     !IS WITHIN THE LIMITS OF THE SPECS
8025 PRINT BP$;                         'OF THE 8020.
8050 WAIT 500
8075 NEXT IZ
8100 PRINT CPOS(3,26);A$;               !THIS IS WHERE WE PRINT THE Z ERROR
8125 A$='0'
8150 RETURN
8175'
8200 ELZ=PZ                             !THIS IS THE SUBR. THAT DOES THE ASCII
8225 IF N$(ELZ)='.' THEN 8400           !ADD. IT LOOKS FOR . AND SKIPS, AND
8250 ITZ=ASCII(N$(ELZ))-48Z+VLZ         'DOES ALL OTHER HOUSE WORK, INCLUDING
8275 I1Z=ABS(ITZ)                       'PRINTING OF THE CHANGED CHARACTER(S)
8300 N$(ELZ)=CHR$(I1Z+48Z)
8325 IF (ITZ>=0Z AND ITZ<9Z) OR (ITZ<=9Z AND ITZ>0Z) THEN 8450
8350 IF ITZ<0Z THEN N$(ELZ)='9'\ PRINT CPOS(2,25Z+ELZ);N$(ELZ);
8375 IF ITZ>9Z THEN N$(ELZ)='0'\ PRINT CPOS(2,25Z+ELZ);N$(ELZ);
8400 ELZ=ELZ-2Z                         !NOTE THIS IS TWO, ONLY BECAUSE THERE
8425 GOTO 8225                          'ARE SPACES IN THE STRING.
8450 PRINT CPOS(2,25Z+ELZ);N$(ELZ);
8475 RETURN
8500'
8525 PRINT @CLZ,C$                      !AND NOW WE GET INTO THE 5100 HANDELER
8550 PRINT S5$;                         'AND OUTPUT THE COMMAND TO THE 5100.
8575 IF X THEN PRINT @CLZ,'$'           !IN THIS ROUTINE WE OUTPUT THE COMMAND
8600 PRINT CPOS(2,25Z+PZ+1Z);A0$;       !TO THE 5100, AND DO ALL THE NECESSARY
8625 WAIT 2000                          'HOUSEWORK REGARDING THE DISPLAY.
8650 PRINT CPOS(2,25);N$(0,,PZ);
8675 P1Z=PZ\A$='0'
8676'
8700 KYZ=KEY                            'HERE WE LOOK FOR WHAT FUNCTION WE HAVE
8725 ON Y%(KYZ) GOTO 8700,8750          'PRESSED.
8750 ON Y2Z(KYZ) GOSUB 8775,9350,9550
8775 ON YOZ(KYZ) GOSUB 8850,9000,9175
8800 A1$=LEFT(A$,11Z)                   !IN THESE TWO LINES WE SEE IF WE ARE IN
8825 IF X THEN 8850 ELSE IF ABS(VAL(A1$)/100)>T(AZ) THEN GOSUB 7975'THE LIMITS.
8850 ON Y1Z(KYZ) GOTO 8700,31220,8875   'IF NOT THEN WE GO TO ERROR SUBR.
8851'
8875 PRINT @CLZ,'S'                     'AS WE EXIT WE PUT 5100 INTO STANDBY
8950 RETURN
8975'
9000 PRINT @CLZ,':?'                    'AND THIS EDITS UP ON THE 5100
9025 INPUT LINE @CLZ,A$                 'AND GETS AND PRINTS THE Z OF ERROR
9050 VLZ=1Z
9075 GOSUB 8200
9100 PRINT CPOS(3,26);A$;
9125 RETURN
9150'
9175 PRINT @CLZ,':?'                    'AND THIS EDITS DOWN ON THE 5100
9200 INPUT LINE @CLZ,A$                 'AND GETS AND PRINTS THE Z OF ERROR
9225 VLZ=-1
9250 GOSUB 8200
9275 PRINT CPOS(3,26);A$;
9300 RETURN
```

```
9325!
9350 PRINT @CL%,'<'                    !THIS MOVES THE EDIT CURSOR LEFT
9375 PRINT CPOS(2,26%+(PZ%-2%));A0$;
9400 IF PZ%>1% THEN IF N$(PZ%-2%)='.' THEN PZ%=PZ%-2%
9425 PZ%=PZ%-2%
9450 IF PZ%<=1% THEN PZ%=1%
9475 PRINT CPOS(2,26%+(PZ%-2%));HL$;
9500 RETURN
9525!
9550 IF PZ%<P1% THEN PRINT @CL%,'>'ELSE 9700 !AND THIS MOVES THE EDIT CURSOR
9575 PRINT CPOS(2,26%+(PZ%-2%));A0$;        !RIGHT.
9600 IF PZ%<P1%-2% THEN IF N$(PZ%+2%)='.' THEN PZ%=PZ%+2%
9625 PZ%=PZ%+2%
9650 IF PZ%>=P1% THEN PZ%=P1%
9675 PRINT CPOS(2,26%+(PZ%-2%));HL$;
9700 RETURN
31010 I%=KEY
31020 I%=KEY\IF I%=0% GOTO 31020
31030 PRINT BP$;
31040 RETURN
31050 !
31200 !********************** EXIT PROGRAM ************************************
31210 !
31220 PRINT CH$;DS$;CPOS(4%,4%);"Return to Instrument Directory ?";
31230 PRINT CPOS(6%,13%);"[YES]  [NO]";
31240 I%=KEY
31250 K%=KEY
31260 IF K%=44% GOTO 32270
31270 IF K%=46% THEN RESTORE 1775\GOTO 1350
31280 GOTO 31250
31290!
32000 !****************** ERROR HANDLER FOR CALIBRATOR ********************
32010 !
32020 !                  PROGRAM IS CALLED "ERCONT"
32030 EZ = ERR                                  ! ASSIGN ERROR NUMBER TO EZ
32040 IF EZ/100% <> 4% THEN GOTO 32250          ! CHECK FOR IEEE ERROR
32050 PRINT CH$;DS$;CPOS(3%,16%);"IEEE-488";
32060 PRINT CPOS(5%,11%);"ERROR ENCOUNTERED";BL$;
32070 WAIT 1000%
32080 PRINT CH$;CPOS(4%,3%);"INSTRUMENTS ARE TO BE CONNECTED BY"
32090 PRINT CPOS(5%,5%);"AN IEEE CABLE TO IEEE PORT 0."
32100 PRINT CPOS(7%,6%);"[ NEXT ]";CPOS(7%,30%);"[ EXIT ]";BL$;
32110 GOSUB 31010                               ! EXIT TO WAIT KEY LOOP
32120 IF I%=59% OR I%=60% THEN RESUME 32270
32130 PRINT CH$;CPOS(4%,8%);"CALIBRATOR SHOULD BE SET TO"
32140 PRINT CPOS(5%,6%);"ADDRESS 3, ( ON REAR PANEL )"
32150 PRINT CPOS(7%,6%);"[ NEXT ]";CPOS(7%,30%);"[ EXIT ]";BL$;
32160 GOSUB 31010                               ! EXIT TO WAIT KEY LOOP
32170 IF I%=59% OR I%=60% THEN RESUME 32270
32180 PRINT CH$;CPOS(4%,3%);"RE-CHECK ALL POWER, CABLE CONNECTIONS"
32190 PRINT CPOS(5%,6%);"AND IEEE ADDRESS SELECTIONS"
32200 PRINT CPOS(7%,6%);"[ RETURN ]";CPOS(7%,30%);"[ EXIT ]";BL$;
32210 GOSUB 31010
32220 IF I%=59% OR I%=60% THEN RESUME 32270
32230 RESTORE 1775
32240 RESUME 1350
32241'
32250 PRINT CH$;DS$;CPOS(5%,9%);"FATAL SOFT/HARD ERROR";
32260 RESUME 32270                              ! CTRL/C ENTRY POINT
32270 PRINT CH$;DS$;CPOS(7%,11%);"RETURNING TO DEMO";BL$;
```

```
32280 IF ERR <> 0% THEN RESUME 32290          ! CLEAR ERROR HANDLER IF ACTIV
32290 OFF ERROR                               ! TURN OFF ERROR
32300 OFF CTRL/C                              ! TURN OFF CTRL/C TRAP
32310 ON ERROR GOTO 32350                     ! TURN ERROR ON FOR EXIT
32320 INIT PORT 0                             ! RESET IEEE INSTRUMENTS
32330 CLEAR @CL%                              ! CLEAR IEEE INSTRUMENTS
32340 GOTO 32370
32350 RESUME 32360
32360 OFF ERROR
32370 RUN"DEMO"
32380 STOP
32767 END
```

3.19 SYSTEMS PROGRAMMING ROM STATEMENT EQUIVALENTS

The following table gives BASIC equivalents for the 9825A System's Programming ROM:

Table 3-13

Systems Programming ROM Statement Equivalents

9825A HPL	1720A BASIC
Statements	
on key "<label>",7 (keyboard interrupt routine enable)	ON KEY GOTO <line No.> (uses operator's CRT Touch-Sensitive Display)
kret(keyboard interrupt routine return)	RESUME
eol(end of line specification) wsc(write serial control word) wsm(write serial mode word)	Not available
rkbd(remote keyboard enable) rkbd II	Use RS 232 Ports for remote keyboard
store	Not needed (string capability is always present)
Functions	
key(key buffer empty)	KEY (operator's CRT Touch-Sensitive Display)
asc(keycode to ASCII conversion	Programmable with "Touch-Sense"
bred(read transfer buffer)	not available
rss(read serial status word)	
nal(next available line	
avm(available memory)	MEM(returns memory space remaining in BYTES)
cln(current line number)	Not available
Miscellaneous	
% % text	! or REM

SECTION 4

1720A PROGRAMMING TIPS

4.1 Programming Pitfalls (GOTCHAS)

The following is a list of commonly made 1720A programming mistakes. By knowing about these ahead of time, you can avoid them.

1. Assignment: Order is reversed from HPL, e.g. A-->B is B=A
2. Relational operators are order dependant
3. IF statement requires a THEN or a GO TO
4. GO TO or GOSUB is executed as soon as it is seen by the program.
5. True truth statements are worth -1 instead of plus 1. (See 22, below)
6. Functions will not execute in the BASIC immediate mode unless they are preceded by PRINT.
7. A=B=C won't work; use A=C\B=C.
8. Implied multiplication is not allowed.
9. Undimensioned variables cannot be allocated in a DIM statement.
10. There is no beep mnemonic. Use PRINT CHR$(7).
11. Relative GO TO or GOSUB is not allowed.
12. Relational comparisons are made to 15 digits instead of 12 digits.
13. Imbedded assignment not allowed.
14. If RETURN is pressed when the cursor is between statements in a line, the portion to the right of the cursor will be transferred to a newly created line immediately below. There is no way to rejoin the two sections except re-entry (EDIT MODE).
15. A line of programming will be stored when the cursor is directed to a higher (or lower) line number. (EDIT MODE).
16. Round brackets are used around dimensions instead of square brackets.

17. ; between statements does not separate them. \ must be used to separate statements. This can be confusing when a semi-colon is used to suppress a line feed.

18. A space is not allowed between GO and SUB for GOSUB.

19. ent A,B,C is different from INPUT A,B,C. In HPL, CONTINUE must be pressed after A and after B and after C. In BASIC, A,B,C may be entered all at once with the two commas, followed by pressing RETURN.

20. Although 1720A arrays are dimensioned from 0 to N; string character positions are dimensioned from 1 to N, i.e., there is no position 0.

21. GOSUB cannot be used in an IF statement unless it is preceded by THEN.

22. False truth statements are worth zero, HOWEVER, the statement IF NOT 0 is only true for -1. For purposes of HPL translation, this means that statements like "if not Z ;..." should always be translated to "IF Z=0...if + 1 is used for true.
The "NOT" operator can be used with flags as long as flags are set to -1 for true and 0 for false, e.g., the statement "IF NOT FG%(1) THEN..."is a valid translation for "if not flg1;...". "IF FG%(1) THEN..." is a valid translation for "if flg1;..." because the statement IF FG%(1) is true for all non-zero values.

23. The 1720A disk is unprotected from a write operation when the write protect notch is uncovered. This is opposite the 9825A cassette and opposite of the 9885 disk.

24. Fluke BASIC does not suport the recording of sections of programs. This can be accomplished, however, by first deleting unwanted lines (DELETE <line No.>-<line No.>) and then recording the program (SAVE 'file name').

    Fluke BASIC does not support the merging of separately recorded programs into one contiguous program. Separate programs may, however, be merged by using the File Utility Program (FUP) via the Merge Utility (<file name>=<file name>,<file name>/M<CR>)

25. OPEN statements should always be closed prior to finishing a program. If a program is stopped due to an error, any open files should be closed, in the immediate mode, prior to running the program again. A temporary file is created on the disk every time an OPEN statement is executed. This temporary file is killed by the CLOSE statement. Failing to CLOSE files will fill the disk with temporary files. These temporary files can be listed in FUP by the command

/E <CR>.

26 The last character in a program line must be a carriage return <CR>, i.e., the RETURN KEY or scrolling the cursor. The resulting character cannot be seen at the end of the line, but it will occupy one character space. If the line is already 80 characters long before it is stored, it will store, but without a <CR>. Future editing of the program will then result in lost sections of lines. Always leave room in a line for the <CR>.

27. After editing a program line it is best to scroll the cursor prior to pressing CTRL C. If there is a syntax error in the line, scrolling the cursor will detect it. If corrections are not made prior to pressing CTRL C, all editing of the line will be lost.

28. Only real and integer variables can be passed by the COM statement. String variables cannot be passed. In order to pass string variables, store them in a Virtual Array.

29. Spaces will be lost in DATA statement strings unless the data is surrounded by quotes. In unquoted strings, leading blanks will be ignored and no commas may appear.

30. Inability to store programs on the Floppy Disk will be experienced when the Electronic Disk is assigned as the system device. Prefixing the file names with "MF0:" will allow access to the Floppy.

4.2 Programming Enhancements
(Also refer to Enhancement Features in Section 2.1)

You can improve your former HPL program by making use of the following enhancements:

1. Integer arithmetic runs faster than floating point.

2. Real time clock

3. Calendar

4. Greek alphabet characters (optional - special)

5. PRINT USING statement (commas allowed)

6. Remark statements

7. More choices for variables; two character variables allow for more descriptive names

8. Line numbers can be assigned in a meaningful manner - see Fluke Convention in Appendix 5C

9. Strings have no length limit except available memory.

10. CRT Display - user defined keys (Touch-Sensitive Display) - Attributes - Graphics (see next 9 pages for examples and programs)

11. Removeable programmer's keyboard - less confusion for operator; less chance of interfering with the program

12. Dual IEEE Ports - Dual buses allow the programmer to place the fast instruments on one bus and the slow instruments on the other bus

13. Dual RS232 Ports - allows controllers to talk back and forth to each other on one port while using the other port to drive a printer

14. Optional Electronic Disk - 4 to 40 times faster than the floppy. Additional 128K Bytes of memory storage per E-Disk

The transparent switch matrix is positioned directly over the CRT display and provides an array of 6 rows of 10 touch-sense areas (See Figure 1). This allows a dynamically redefinable keyboard function for use during program execution. The touch-sense has different characteristics from the keyboard:

- Automatic repeat when one or more keys are held

- No-key rollover

Each sensitive key area covers three character locations in the double character size display mode (2 rows and 6 columns in the single character size). If more than one key is depressed, only one key will be recognized.

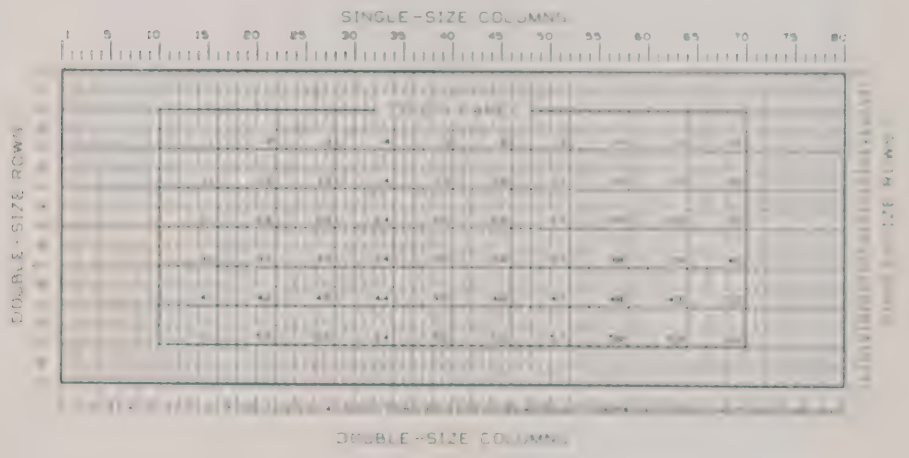
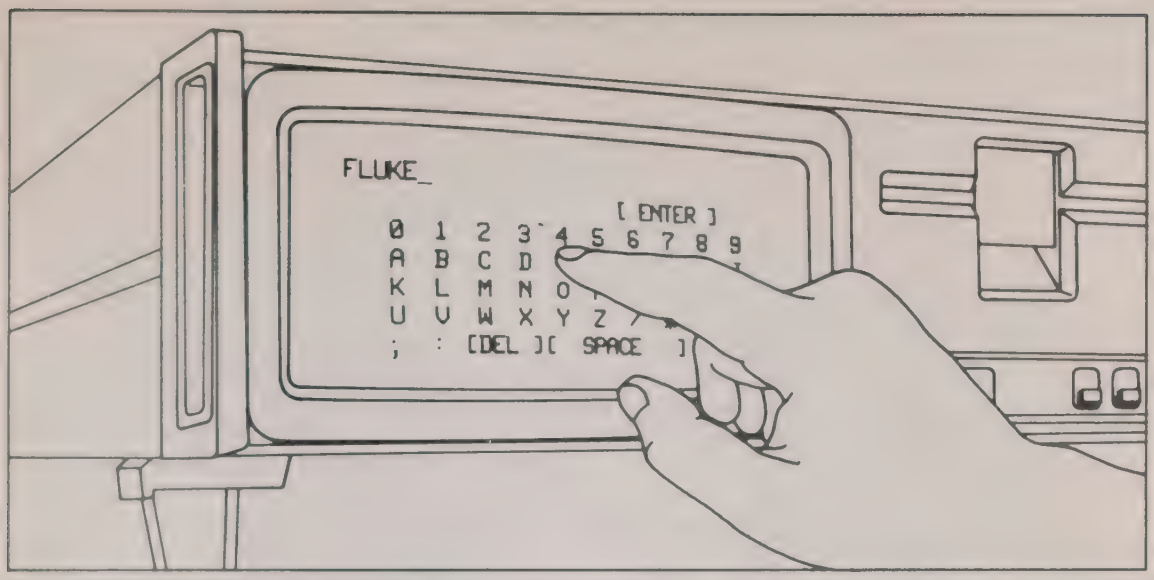
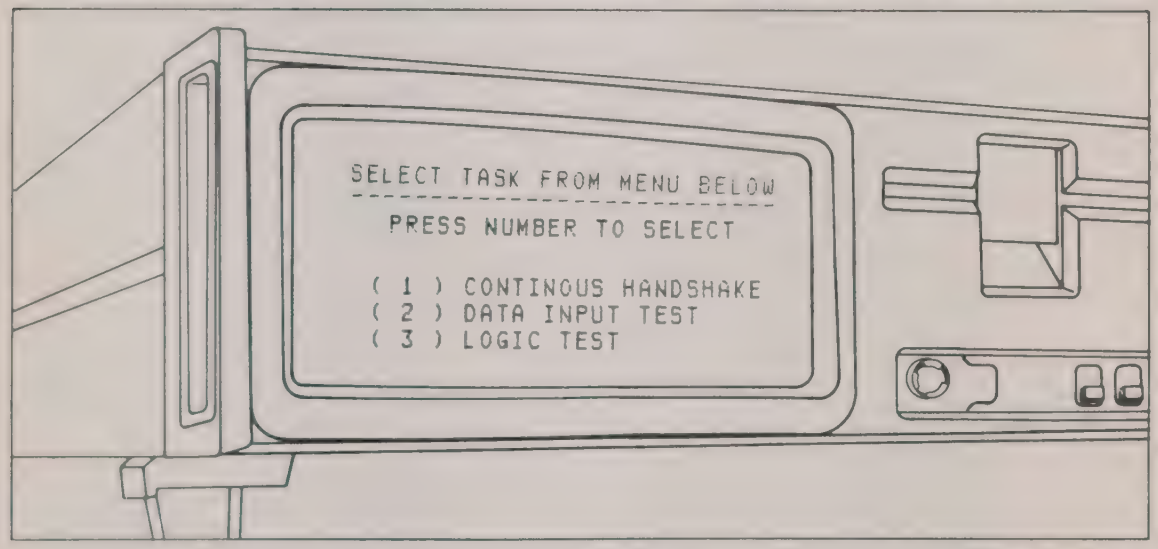


Figure 4-1





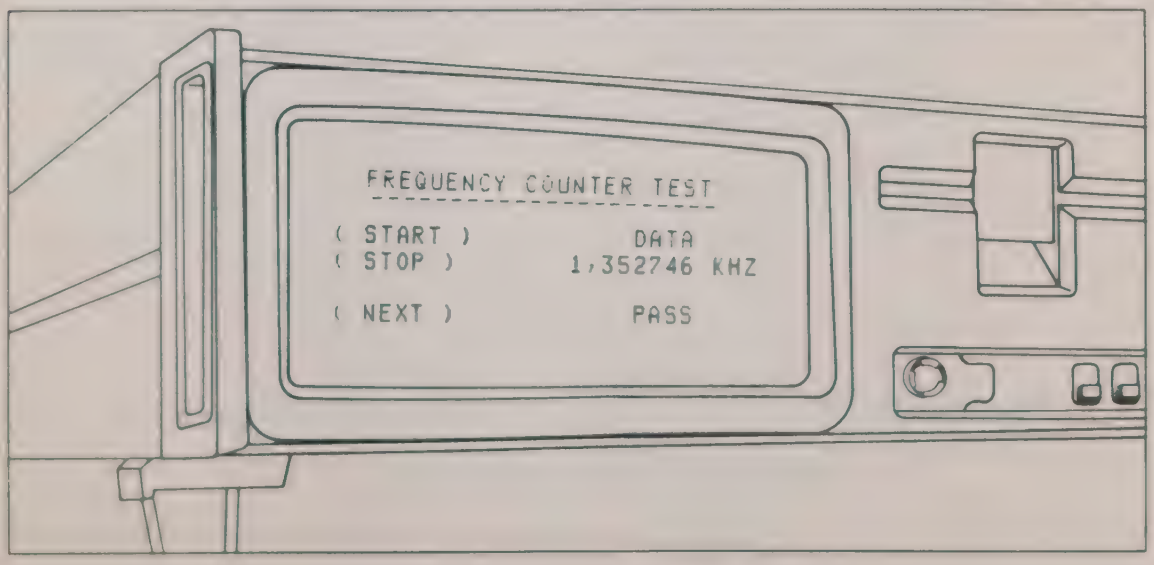


Figure 4-2 Operator Display

Program 22 - Basic

```
1  ' "TOUCH",              TOUCH SENSE DEMO
7  ' VERSION 1.0, 16 NOVEMBER 1979
15 '
20 '*************************************************************
30 '
100 ' THIS PROGRAM DEMONSTRATES THE USE OF TOUCH SENSE WITH DISPLAYED ENTRY
110 ' CHOICES.  THE USER CAN ENTER ANY PROMPT DESIRED.  FOR DEMONSTRATION
120 ' PURPOSES ONLY, PROMPTS ARE ENTERED FROM THE PROGRAMMER'S KEYBOARD.
130 ' NORMALLY A PROMPT WOULD BE PART OF THE PROGRAM.  THE 40 CHARACTER LINE
140 ' IS DIVIDED UP BETWEEN THE PROMPT AND THE MESSAGE.  NO. OF CHARACTERS
150 ' AVAILABLE FOR THE MESSAGE = 40-3-NO. CHARACTERS IN THE PROMPT.
160 '
170 ' THE TOUCH SENSE FEATURE OPERATES AS FOLLOWS:  THE OPERATOR TOUCHES
180 ' ONE OF THE KEYPADS ON THE 10 X 6 KEYPAD ARRAY ON THE FACE OF THE CRT.
190 ' THE OPERATOR KNOWS WHICH KEYPAD TO TOUCH BY SELECTING FROM THE CRT
200 ' DISPLAYED OVERLAY.  THE ARRAY VALUE ( 1 THROUGH 60) OF THE TOUCHED
210 ' KEYPAD IS STORED BY THE SYSTEM INTO THE SYSTEM VARIABLE "KEY".
220 '
230 '*************************************************************
240 '
590 '                    DISPLAY MODES
595 '
600 ES$ = CHR$(27)+"["           ! ESCAPE INITIALIZATION
610 NS$ = ES$ + "p"              ! NORMAL SIZE CHARACTERS
620 DS$ = ES$ + "1p"             ! DOUBLE SIZE CHARACTERS
630 OG$ = ES$ + "2p"             ! DISABLES GRAPHICS MODE
640 EG$ = ES$ + "3p"             ! ENABLES GRAPHICS MODE
650 US$ = ES$ + "4m"             ! UNDERSCORE
660 BE$ = ES$ + "5m"             ' BLINK
670 AO$ = ES$ + "m"              ' ATTRIBUTES OFF
680 IV$ = ES$ + "7m"             ' REVERSE VIDEO
690 HL$ = ES$ + "1m"             ' INCREASED INTENSITY
692 '
695 '*************************************************************
696 !
700 '                   ERASE MODES
705 '
710 ED$ = ES$ + "J"              ' ERASE FROM ACTIVE POSITION TO END OF SCREEN
720 EU$ = ES$ + "1J"             ' ERASE FROM START OF SCREEN TO ACTIVE POSITI
730 EP$ = ES$ + "2J"             ' ERASE PAGE
740 EE$ = ES$ + "0K"             ' ERASE FROM ACTIVE POSITION TO END OF LINE
750 EF$ = ES$ + "1K"             ' ERASE FROM FIRST OF LINE TO ACTIVE POSITION
760 EL$ = ES$ + "2K"             ' ERASE LINE
770 EC$ = ES$ + "D " + LS$ + "D" ' ERASE CHARACTER
780 '
790 RS$ = ES$ + "0;2p"           ' RESET SCREEN
800 '
805 '*************************************************************
820 '
830 '                   VARIABLE DEFINITIONS
840 '
850 ' A1Z - LENGTH OF A$ - NO. OF CHARACTERS IN THE MESSAGE
860 ' A$ - STORAGE FOR THE MESSAGE
870 ' KZ - KEY   ARRAY VALUE SUBSCRIPT FOR TOUCH SENSE KEYPAD
880 ' P$ = PROMPT
890 ' Y% - LENGTH OF PROMPT + 2 SPACES + COLON
897 '
900 DIM A$(60)  ' DIMENSION STRING TO CONTAIN TOUCH SENSE OUTPUT CHARACTERS
910 '
1000 '*************************************************************
```

```
1010 !
1020 !          ASSIGN DISPLAYED TOUCH SENSE CHARACTERS TO A$ ARRAY
1030 !
1040 !            BLANK CHARACTERS FOR SPACE ENTRY
1050 FOR I% = 55% TO 58% \ A$(I%) = CHR$(32%) \ NEXT I%
1060 !
1070 !            NUMBERS 0 THROUGH 9
1080 FOR I% = 11% TO 20% \ A$(I%) = CHR$(37+I%) \ NEXT I%
1090 !
1100 !           LETTERS A THROUGH Z
1110 FOR I% = 21% TO 46% \ A$(I%) = CHR$(44+I%) \ NEXT I%
1120 !
1130 !                                    CHARACTERS
1140 A$(47%) = CHR$(47%)         !          /
1150 A$(48%) = CHR$(42%)         !          *
1160 A$(49%) = CHR$(44%)         !          ,
1170 A$(50%) = CHR$(46%)         !          .
1180 A$(51%) = CHR$(59%)         !          ;
1190 A$(52%) = CHR$(58%)         !          :
1200 A$(59%) = CHR$(37%)         !          %
1210 A$(60%) = CHR$(45%)         !          -
1220 !
2000 !*************************************************************************
2010 !
2020 !                       MAIN PROGRAM
2030 !
2045 PRINT RS$;             ! RESET SCREEN
2050 PRINT "ENTER PROMPT YOU WOULD LIKE TO SEE DISPLAYED THEN PRESS RETURN"
2060 INPUT P$
2070 X%=LEN(P$)+3%          ! MEASURE LENGTH OF PROMPT +  2 SPACES AND :
2080 !
2085 A$ =""  \ A1%=0%       ! INITIALIZE STRING WHICH WILL RECEIVE MESSAGE
2090 GOSUB 10040            ! GO GET THE TOUCH SENSE INPUT
2100 GOTO 2045              ! GO GET ANOTHER PROMPT
2110 !
10000 !************************************************************************
10010 !
10020 !           TOUCH SENSE:  DISPLAY AND ENTRY
10030 !
10040 PRINT DS$;DG$;P$;": MESSAGE"  ! INITIALIZE SCREEN AND DISPLAY PROMPT
10050 !
10060 K% = KEY                      ! DEFINE TOUCH SENSE VARIBLE
10070 !
10080 !                 DISPLAY TOUCH SENSE INPUT ARRAY
10090 !
10100 PRINT CPOS(2,24);"[ ENTER ]"       ! CURSOR POSITION(ROW2,COLUMN 24)
10110 PRINT "      0  1  2  3  4  5  6  7  8  9 "
10120 PRINT "      A  B  C  D  E  F  G  H  I  J "
10130 PRINT "      K  L  M  N  O  P  Q  R  S  T "
10140 PRINT "      U  V  W  X  Y  Z  /  *  ,  . "
10150 PRINT "      ;  :  [DEL ][  SPACE   ] %  - ";
10160 PRINT ES$;"H";                    ! HOME THE CURSOR
10170 !
10180 !************************************************************************
10190 !
10200 !             GET NUMBER OF KEYPAD THAT WAS TOUCHED
10210 !
10220 !                  RECEIVE TOUCH SENSE INPUT
10230 K% = KEY \ IF K% = 0% THEN 10230    ! HAS KEYPAD BEEN TOUCHED?
10240 PRINT CPOS(1,X%);EL$; \ GOTO 10270  ! GET RID OF "MESSAGE"
```

```
10250 KZ = KEY \ IF KZ = 0Z THEN 10250    ! HAS KEYPAD BEEN TOUCHED?
10260 !
10270 PRINT CHR$(7);              ! AUDIBLE FEEDBACK (ACKNOWLEDGE ENTRY)
10280 !
10290 '*************************************************************************
10300 !
10310 !                  WHAT KIND OF ENTRY IS IT?
10320 '
10330 IF KZ = 53Z OR KZ = 54Z GOTO 10500  ! TOUCH SENSE KEYPAD FOR [DEL ]
10340 IF KZ < 10Z AND KZ > 6Z GOTO 10610  ! TOUCH SENSE KEYPAD FOR [ ENTER ]
10350 !
10360 '*************************************************************************
10370 !
10380 !                    CHARACTER ENTRY AND MESSAGE UPDATE
10390 !
10400 IF LEN(A$)+LEN(A$(KZ))>38Z-XZ+2Z GOTO 10250 ! TEST FOR TOO LONG AN ENTRY
10410 A$ = A$ + A$(KZ)                    ! CONCATINATE NEWELY ENTERED CHARACTER
10420 A1Z = LEN (A$)                      ! UPDATE LENGTH OF A$
10430 PRINT A$(KZ);                       ! DISPLAY UPDATED MESSAGE
10440 GOTO 10250                          ! GO GET ANOTHER TOUCH SENSE INPUT
10450 '
10460 '*************************************************************************
10470 !
10480 !              DELETE THE LAST CHARACTER ROUTINE
10490 !
10500 A$ = LEFT(A$,LEN(A$)-1Z)            ! REMOVE LAST CHARACTER ENTRY FROM A$
10510 IF A1Z<=0Z THEN 10250               ! HAS A$ BEEN REDUCED TO NULL STRING?
10520 PRINT EC$;                          ! REMOVE LAST CHARACTER ENTRY FROM
10530                                     ! MESSAGE ON SCREEN
10540 A1Z=A1Z-1Z                          ! UPDATE LENGTH OF A$
10550 GOTO 10250                          ! GO GET ANOTHER TOUCH SENSE INPUT
10560 '
10570 '*************************************************************************
10580 !
10590 !          MESSAGE IS COMPLETE ( [ ENTER ] HAS BEEN TOUCHED )
10600 '
10610 PRINT KS$                           ! RESET SCREEN, ENTRY IS COMPLETE
10620 RETURN                              ! GO GET ANOTHER PROMPT
32767 END
```

Program 23 - Basic

```
1   ! "ATDEMO",   DISPLAY ATTRIBUTES DEMO
7   ! VERSION 1.0, 19 NOVEMBER 1979
15  !
100 ! THIS PROGRAM DEMONSTRATES THE USE OF THE VIDEO ATTRIBUTES.  THE
110 ! ATTRIBUTES ARE DEMONSTRATED IN THIS ORDER:
120 !
130 !        1. CPOS(ROW NO., COLUMN NO.) ---- CURSOR POSITIONING
140 !        2. HIGHLIGHTED CHARACTERS
150 !        3. BLINKING CHARACTERS
160 !        4. UNDERSCORED CHARACTERS
170 !        5. INVERSE VIDEO CHARACTERS
180 !        6. COMBINATION OF BLINKING, UNDERSCORE AND INVERSE VIDEO
190 !        7. RETURN TO NORMAL CHARACTERS
200 !        8. DOUBLE SIZE CHARACTERS
205 !        9. RETURN TO NORMAL SIZE CHARACTERS
210 !       10. RESET SCREEN
220 !
230 ! CPOS IS DEMONSTRATED BY DRAWING A PARTIAL CIRCLE WITH THE LETTERS "JF".
240 ! THE OTHER ATTRIBUTES ARE DEMONSTRATED ONE AT A TIME IN SUCCESSION WITH
250 ! THE NAME "JOHN FLUKE".  BY TOUCHING THE SCREEN THE OPERATOR IS ABLE TO
260 ! STEP THROUGH THE ATTRIBUTES IN THE ABOVE ORDER.
270 !
280 ! BE SURE TO NOTE THAT THE ATTRIBUTES MUST BE PROGRAMED TO TURNED OFF
290 ! AT THE POSITION WHERE YOU WANT THEM TO TURN OFF, PRIOR TO PROGRAMMING
300 ! THESE SAME ATTRIBUTES TO TURN ON.  AT FIRST THIS MAY SEEM BACKWARDS,
310 ! BUT THE LOGIC OF DOING THIS WILL BECOME APPARENT AFTER YOU HAVE DONE
320 ! SOME PROGRAMMING WITH THE ATTRIBUTES.  AN EXAMPLE OF THIS APPEARS IN
330 ! LINE 2190.
340 !
350 !*****************************************************************************
360 !
500 !                   DISPLAY MODES
505 !
510 ES$ = CHR$(27)+"["  !             ESCAPE INITIALIZATION
520 NS$ = ES$ + "p"     !             NORMAL SIZE CHARACTERS
530 DS$ = ES$ + "1p"    !             DOUBLE SIZE CHARACTERS
540 DG$ = ES$ + "2p"    !             DISABLES GRAPHICS MODE
550 EG$ = ES$ + "3p"    !             ENABLES GRAPHICS MODE
560 US$ = ES$ + "4m"    !             UNDERSCORE
570 BE$ = ES$ + "5m"    !             BLINK
580 AO$ = ES$ + "m"     !             ATTRIBUTES OFF
590 IV$ = ES$ + "7m"    !             REVERSE VIDEO
600 HL$ = ES$ + "1m"    !             INCREASED INTENSITY
602 RS$ = ES$ + "0;2p"  !             RESET SCREEN
603 CB$ = ES$ + "1;4;5m"            ! COMBINATION
605 !
607 !*****************************************************************************
608 !
610 !                   ERASE MODES
615 !
620 ED$ = ES$ + "J"     !           ERASE FROM ACTIVE POSITION TO END OF SCREEN
630 EU$ = ES$ + "1J"    !           ERASE FROM START OF SCREEN TO ACTIVE POSITI
640 EP$ = ES$ + "2J"    !           ERASE PAGE
650 EE$ = ES$ + "0K"    !           ERASE FROM ACTIVE POSITION TO END OF LINE
660 EF$ = ES$ + "1K"    !           ERASE FROM FIRST OF LINE TO ACTIVE POSITION
670 EL$ = ES$ + "2K"    !           ERASE LINE
680 EC$ = ES$ +"D " + ES$ + "D"   ! ERASE LAST CHARACTER
690 !
700 BP$=CHR$(7)+CHR$(7)+CHR$(7)+CHR$(7)     ' LONG BEEP
720 BL$=ES$ + "16;H"    !           CURSOR TO BOTTOM LEFT
```

```
730 '
2000 '*************************************************************************
2010 '
2020 ' MAIN PROGRAM
2030 '
2040 '                    CIRCLE GENERATOR ( CPOS DEMO )
2050 '
2060 PRINT RS$                          ' RESET SCREEN
2070 FOR A=126 TO 414 STEP 9            ' FROM 126 DEG. TO 414 DEG.
2080 IF A=243 THEN A=297 \ GOTO 2150    ' SKIP BOTTOM OF CIRCLE
2090 CV=A*PI/180                        ' CONVERSION: DEGREES TO RADIANS
2100 X=COS(CV)*17                       ' HORIZONTAL COORDINATE
2110 Y=SIN(CV)*7.2                      ' VERTICAL COORDINATE
2120 X=INT(X+.5)+40%                    ' ROUND X AND RELOCATE CENTER
2130 Y=-INT(Y+.5)+9%                    ' ROUND Y AND RELOCATE CENTER
2140 PRINT CPOS(Y,X);"JF";              ' DISPLAY "JF" AT X,Y COORDINATES
2150 NEXT A
2160 '
2170 '                    CHARACTER ATTRIBUTES
2180 '
2190 PRINT CPOS(9,45);AO$;              ' TURN ATTRIBUTES OFF IMMEDIATELY
2200                                    ' AFTER THE "E" IN "JOHN FLUKE"
2210 PRINT CPOS(9,35);"JOHN FLUKE";     ' DISPLAY "JOHN FLUKE"
2220 GOSUB 10040
2230 PRINT CPOS(9,34);HL$;              ' HIGHLIGHT
2240 GOSUB 10090
2250 PRINT CPOS(9,34);BE$;              ' BLINK ENABLE
2260 GOSUB 10090
2270 PRINT CPOS(9,34);US$;              ' UNDERSCORE
2280 GOSUB 10090
2290 PRINT CPOS(9,34);IV$;              ' INVERSE VIDEO
2300 GOSUB 10090
2310 PRINT CPOS(9,34);CB$;              ' COMBINATION OF IV$, BE$ AND US$
2320 GOSUB 10090
2330 PRINT CPOS(9,34);AO$;              ' TURN ATTRIBUTES OFF BUT CONTINUE
2340                                    ' DISPLAYING "JOHN FLUKE" NORMALLY
2350 GOSUB 10090
2360 PRINT DS$;CPOS(5,15);"JOHN FLUKE";' DOUBLE SIZE CHARACTERS
2370 GOSUB 10090
2380 PRINT NS$;CPOS(9,35);"JOHN FLUKE";' NORMAL SIZE CHARACTERS
2390 GOSUB 10090
2400 PRINT RS$;
2410 END
2420 '
10000 '*************************************************************************
10010 '
10020 '                      TOUCH SENSE PROMPT
10030 '
10040 PRINT CPOS(13,2);"TOUCH CENTER"
10050 PRINT CPOS(14,3);"OF SCREEN"
10060 PRINT CPOS(15,2);"TO CONTINUE"
10070 '
10080 '                      TOUCH SENSE INPUT
10090 PRINT BL$;
10100 K2=KEY
10110 K2=K2*
10120 IF K2=0% GOTO 10100
10130 PRINT BP$;
10140 RETURN
32767 END
```

Program 24 - Basic

```
1   ! "GRDEMO",   GRAPHICS DEMO
7   ! VERSION 1.0, 19 NOVEMBER 1979
15  !
30  !*************************************************************************
40  !
100 !  THIS PROGRAM DEMONSTRATES THE USE OF THE GRAPHIC CHARACTERS USING
110 !  NORMAL SIZE GRAPHICS.
230 !
240 !*************************************************************************
250 !
500 !                   DISPLAY MODES
505 !
510 ES$ = CHR$(27)+"[" !             ESCAPE INITIALIZATION
520 NS$ = ES$ + "p"    !             NORMAL SIZE CHARACTERS
530 DS$ = ES$ + "1p"   !             DOUBLE SIZE CHARACTERS
540 DG$ = ES$ + "2p"   !             DISABLES GRAPHICS MODE
550 EG$ = ES$ + "3p"   !             ENABLES GRAPHICS MODE
560 US$ = ES$ + "4m"   !             UNDERSCORE
570 BE$ = ES$ + "5m"   !             BLINK
580 AO$ = ES$ + "m"    !             ATTRIBUTES OFF
590 IV$ = ES$ + "7m"   !             REVERSE VIDEO
600 HL$ = ES$ + "1m"   !             INCREASED INTENSITY
602 RS$ = ES$ + "0;2p" !             RESET SCREEN
603 CB$ = ES$ + "1;4;5m"           ! COMBINATION
605 !
607 !*************************************************************************
608 !
610 !                   ERASE MODES
615 !
620 ED$ = ES$ + "J"    !          ERASE FROM ACTIVE POSITION TO END OF SCREEN
630 EU$ = ES$ + "1J"   !          ERASE FROM START OF SCREEN TO ACTIVE POSITI
640 EP$ = ES$ + "2J"   !          ERASE PAGE
650 EE$ = ES$ + "0K"   !          ERASE FROM ACTIVE POSITION TO END OF LINE
660 EF$ = ES$ + "1K"   !          ERASE FROM FIRST OF LINE TO ACTIVE POSITION
670 EL$ = ES$ + "2K"   !          ERASE LINE
680 EC$ = ES$ +"D " + ES$ + "D"   ! ERASE LAST CHARACTER
690 !
700 BP$=CHR$(7)+CHR$(7)+CHR$(7)+CHR$(7)     ! LONG BEEP
710 !
720 BL$=ES$ + "16;H"
730 !
1000 !************************************************************************
1010 !
1020 ! MAIN PROGRAM
1030 !
1040 !                    GRAPHICS GENERATOR
1050 !
1060 PRINT RS$                        ! RESET SCREEN
1070 !
1080 PRINT EG$                        ! ENABLE GRAPHICS
1090 !
1100 ! T$ IS THE TOP HORIZONTAL LINE, INCLUDING TOP CORNERS
1110 ! I$ IS ANY INSIDE HORIZONTAL LINE, AND VERTICAL DASHES
1120 ! S$ FOR THE SPACES BETWEEN THE HORIZONTAL LINES, AND VERTICAL DASHE
1130 ! B$ IS THE BOTTOM LINE, BOTTOM CORNERS, AND VERTICAL DASHES
1140 ! THE CHARACTER "8" IS A HORIZONTAL DASH
1150 ! THE CHARACTER "9" IS A VERTICAL DASH
1160 ! THE CHARACTERS "0","1", "2", AND "3" FORM THE CORNERS
1170 ! THE CHARACTER "7" IS SIMILAR TO A"+" CHARACTER
1180 ! CPOS(ROW,COLUMN) IS CURSOR POSITIONING FUNCTION
```

```
1190 !
1200 !                    CONSTRUCT GRID
1210 !
1220 T$="1888888888
```

Table 4-1

Attributes and Graphics

Type	Action	Sequence	Print Statement
CURSOR CONTROLS	Up n lines	ESC nA	PRINT CHR$(27);" nA";
	Down n lines	ESC nB	PRINT CHR$(27);" nB";
	Right n columns	ESC nC	PRINT CHR$(27);" nC";
	Left n columns	ESC nD	PRINT CHR$(27);" nD";
	Direct line;column	ESC l;cH	PRINT CHR$(27);" l;cH";
	(or use CPOS function)		PRINT CPOS(l,c);
ERASING	To line end	ESC K	PRINT CHR$(27);" K";
	From line start	ESC 1K	PRINT CHR$(27);" 1K";
	All of line	ESC 2K	PRINT CHR$(27);" 2K";
	To display end	ESC J	PRINT CHR$(27);" J";
	From display start	ESC 1J	PRINT CHR$(27);" 1J";
	All of display	ESC 2J	PRINT CHR$(27);" 2J";
ENHANCEMENTS	Enhancements off	ESC m	PRINT CHR$(27);" m";
	High intensity	ESC 1m	PRINT CHR$(27);" 1m";
	Underline	ESC 4m	PRINT CHR$(27);" 4m";
	Blinking	ESC 5m	PRINT CHR$(27);" 5m";
	Reverse image	ESC 7m	PRINT CHR$(27);" 7m";
MODES	Normal size	ESC p	PRINT CHR$(27);" p";
	Double size	ESC 1p	PRINT CHR$(27);" 1p";
	Graphics OFF	ESC 2p	PRINT CHR$(27);" 2p";
	Graphics ON	ESC 3p	PRINT CHR$(27);" 3p";
PRINTER FUNCTIONS	Line feed	ESC D	PRINT CHR$(27);" D";
	Reverse line feed	ESC M	PRINT CHR$(27);" M";
	Next line	ESC E	PRINT CHR$(27);" E";

4.3 Programming Shortcuts

Here are some time saving hints for better 1720A programming:

1. Duplication of a line (EDIT MODE):

   a. Place the cursor under line number of the line which is to be duplicated
   b. Type in new line number; delete old line number
   c. Store line by moving cursor up or down
   d. Simultaneously Press: CTRL C
   e. Type EDIT
   f. Press RETURN
   g. All lines are displayed

2. The NUM$ function adds a leading and trailing space to the number when it is strung. These spaces can be stripped off. See the example below:

```
A$=NUM$(5) ! A$="b5b" where b is a space

A$=MID(A$,2,1) ! A$="5"
```

3. Use integer arithmetic wherever possible instead of floating point. Integer arithmetic is faster and more accurate when used with relational operators.

4. The Command File is most easily programmed via the File Utility Program (FUP).

   a. Enter CONSOLE MONITOR by simultaneously pressing CRTL P
   b. When the "#" is displayed, enter FUP by typing F and pressing RETURN
   c. Program the Command File as follows:

      (1) Type COMMND.SYS=KB0:<CR>
      (2) Type desired commands (examples follow):

set time	T <CR>
enter FUP	F <CR>
extended listing	/E <CR>
list this Command File	KB1:=COMMND.SYS<CR> (could also be any Basic Program or Data File)
exit FUP	/X
enter Basic	B
run a program called DEMO	RUN "<DEMO>"

      (3) Simultaneously Press CTRL Z

   d. For additional information about the Command File see Section 2.4.1

5. Program line which is too long may be easily split and the latter portion placed on a new subsequent line:

   a. Place cursor between statements where split is to occur
   b. Delete un-needed statement separator \
   c. Press RETURN
   d. Type in new line number

4.4 Obtaining A Printed Program Listing

RS 232 Printer:

Basic Immediate Mode

1. Select an RS232 port and connect the printer.

2. Type: SAVE "KB1:"

   or SAVE "KB2:"

3. Press: RETURN

File Utility Program

Be aware of the fact that when you leave the Basic Mode in preparation to entering the File Utility Program, main memory is erased. Programs must be saved on the disk prior to leaving the Basic Mode.

1. Exit the Basic Mode by simultaneously pressing CTRL and P.
2. The controller is now in the Console Monitor mode. You should see a "#".
3. Enter into FUP by typing F and pressing RETURN.
4. When * is displayed--
   (1) Type KB1:=file name
   (2) Press RETURN

Note: Only ASCII files can be listed in FUP

IEEE-488 Printer

Use the following program which appears on the next page.

Program 19 -Basic

```
10 ' PROGRAM     : LIST.IEZ
20 '
30 ' REVISION    : 1.0       DATE  : 26/10/79
40 '
50 ' DESCRIPTION : LISTS A PROGRAM ON AN IEEE-488 PRINTER
60 '
70 ' IEEE ADDRESS:  PRINTER MUST BE SET TO AN ADDRESS OF 1
80 '
90 '
100                          ON ERROR GOTO 320
110 '
120 '                        COMMON VARIABLES
130 '
140 ES$ = CHR$(27%) + "["                              ESCAPE SEQUENCE
150 DS$ = ES$ + "1p"                                   DOUBLE SIZE CHARACTERS
160 PRZ = 1%                                           IEEE ADDRESS
170 TIMEOUT 3000 <- 20000
180 PRINT DS$;                                         RESET SCREEN
190 INIT PORT 0 \ CLEAR @ PRZ                          INITIALIZE PRINTER
200 PRINT ES$;'4;HENTER FILENAME TO BE LISTED'         PROMPT
210 INPUT A$                                           STORE FILENAME
220 CLOSE 1 \ OPEN A$ AS FILE 1                        OPEN FILE FOR PROGRAM
230 PRINT @ PRZ,CHR$(12%);CHR$(13% )                   TOP OF FORM
240 FOR I% = 1% TO 60%                                 FOR ALL LINES
250 INPUT LINE #1, A$                                  FOR EACH LINE
260 PRINT @ PRZ, A$;CHR$(13%)                          PRINT EACH LINE
270 NEXT I%                                            NEXT LINE
280 GOTO 230
290 '
300'                    ERROR HANDLER
310'
320 IF ERR = 305 THEN RESUME 350                       FILE DOESN'T EXIST
330 IF ERR = 307 THEN RESUME 370                       END OF FILE
340 PRINT 'ERROR';ERR;'@ LINE';ERL; \ RESUME 370
350 PRINT ES$;'6;HFILE DOESN'T EXIST - TRY AGAIN'; \ GOTO 200
360 '
370 CLOSE 1
380 END
```

5 APPENDICES

5A Algorithm To Use One Dimensional Array For a Multidemensional Array

Two Dimensional Array

```
DIM V (Amax * Bmax) ! 1<=I<=A and 1<=J<=B

V(I,J)=V((I-1)*B+J)
```

Three Dimensional Array

```
DIM V (Amax * Bmax * Cmax) ! 1<=I<=A and 1<=J<=B and 1<=K<=C

V(I,J,K)=V((I-1)*B*C+(J-1)*C+K)
```

Four Dimensional Array

```
DIM V (Amax *Bmax *Cmax *Dmax)

        1<=I<=A and 1<=J<=B and 1<=K<=C and1<=L<=D

V (I,J,K,L)=V((I-1)*B*C*D+(J-1)*C*D+(K-I)*D+L)
```

Five Dimensional Array

```
DIM V (Amax *Bmax *Cmax *Dmax *Emax)

        1<=I<=A  and  1<=J<=B  and  1<=K<=<=C  and 1<=L<=D  and
1<=M<=C

V(I,J,K,L,M)=V(I-1)*B*C*D*E+(J-1)*C*D*E+(K-1)*D*E+(L-1)*E+M)
```

General Case (an array of Dn dimensions)

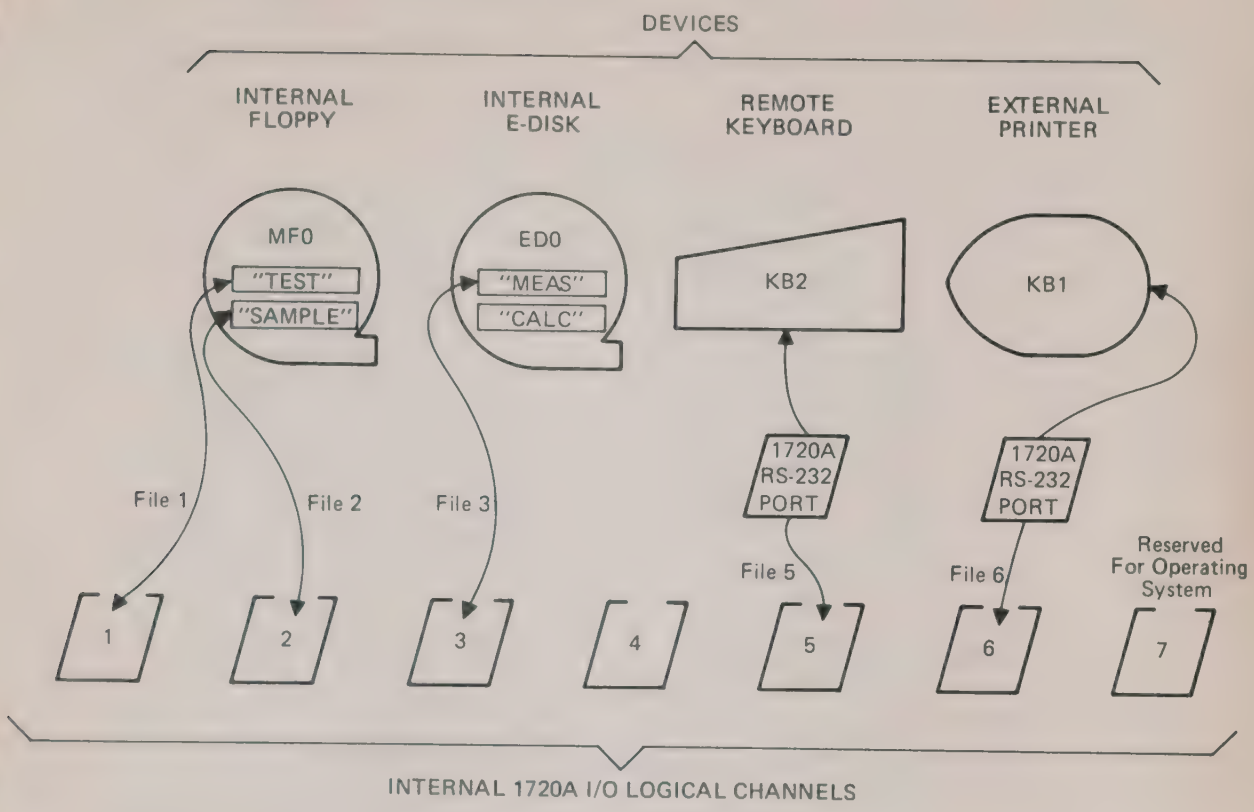
```
DIM (D    *D  *D ...*D)
      1    2   3     n

where 1<= S  <=D  and  1<=S   <=D  and 1=<S  <=D  and ...
          1    1          2     2         3    3
and 1<=S   <=D
       n     n
```

$$V(S_1, S_2, S_3 \ldots S_n) = ((S_1 - 1) * D_2 * D_3 \ldots * D_n + (S_2 - 1) * D_3 \ldots * D_n + (S_3 - 1) \ldots * D_n + \ldots + (D_{n-1} - 1) * D_n + S_n)$$

Note: % signs have been intentionally left out for clarity.

DEVICES, LOGICAL CHANNELS, FILES AND PORTS



The configuration above can be accomplished using the following statments:

	Device No.	File Name	Logical Channel No.
<line No.>	OPEN	"MF0:TEST" AS NEW FILE	1
<line No.>	OPEN	"MF0:SAMPLE" AS NEW FILE	2
<line No.>	OPEN	"ED0:MEAS" AS NEW FILE	3
<line No.>	CLOSE		4
<line No.>	OPEN	"KB2:" AS FILE	5
<line No.>	OPEN	"KB1:" AS FILE	6

OPEN Statements are used to assign devices and file names to logical channels.

PRINT statements are used to write data on peripherals

INPUT and INPUT LINE statements are used to read data from peripherals

5C Programming Conventions Used In
Fluke Instrument Demo Software

Line #		Contents
1	!	Program File Name
2	!	
3	!	Program series
4	!	Program of
5	!	
6	!	Revision No. Program Change Function. Program Change Operational
7	!	Revision Date-Time
8	!	
9	!	Author
10-99	!	Modification History Who, What, When, Why
100-299	!	General Functional Description
300-499	!	Function and Subroutine Descriptions
500-599	!	I/O Channel Identification
600-899		Variables Definitions
900-949		DIM Statements - to max required
950-999		COM Statements
1000-1999		Data Statements
2000		Program Start
2002		ON Error Trap
2010-2020		Header Print
2030-2099		Define Constants
9000-9999		Program Exit Routine
10000-14999		Sub Routines

15000-18999	Defined Functions
19000-19999	Error Handler
20000-29999	Standard Routines
32765	STOP
32766	KILL'FILENAME.EXTENSION'\SAVE'FILENAME'
32767	END

5D Basic Variables Convention

A	A $ temp string for CFI Data Transfer
B	B $ = CHR$(27) + '['
C	
D	
E	E $ Error Handler - Fatal Error Description
F	Flag
G	
H	
I	
J	
K	K % = Key Response from Touch Panel
L	L $ = Misc. Loop Variable
M	
N	
O	
P	
Q	
R	
S	Scratch Temporarily
T	
U	
V	
W	
X	
Y	Subscript Integers
Z	

```
100 '
110 '            BORROW VARIABLES FOR DISPLAY FUNCTIONS
120 '
130 '
140 '
150 ES$  = CHR$(27%) + "[" '      ESCAPE AND LEFT BRACKET
160 '
170 '                  BEEP
180 '
190 BP$ = CHR$(7%)
200 '
210 '                  CLEAR SCREEN AND HOME
220 '
230 CH$ = ES$ + "2J" + ES$ + ";H"
240 '
250 '                  RESET SCREEN
260 '
270 RS$ = ES$ + "2J" + ES$ + "0;2p" + ES$ + ";H"
280 '
290 '                  CURSOR TO BOTTOM LEFT
300 '
310 BL$ = ES$ + "16;H"
320 '
330 '                  DISPLAY MODES
340 '
350 NS$ = ES$ + "p"    '          NORMAL SIZE CHARACTERS
360 DS$ = ES$ + "1p"   '          DOUBLE SIZE CHARACTERS
370 DG$ = ES$ + "2p"   '          DISABLES GRAPHICS MODE
380 EG$ = ES$ + "3p"   '          ENABLES GRAPHICS MODE
390 DK$ = ES$ + "5p"   '          DISABLE KEYBOARD
400 US$ = ES$ + "4m"   '          UNDERSCORE
410 EB$ = ES$ + "5m"   '          ENABLE BLINK
420 AO$ = ES$ + "m"    '          ATTRIBUTES OFF
430 IV$ = ES$ + "7m"   '          INVERSE VIDEO
440 HL$ = ES$ + "1m"   '          HIGHLIGHT (INCREASED INTENSITY)
450 HU$ = ES$ + "1;4m" '          HIGHLIGHT AND UNDERSCORE
460 '
470 '                  CURSOR CONTROLS
480 '
490 CU$ = ES$ + "A"    '          CURSOR UP
500 CD$ = ES$ + "B"    '          CURSOR DOWN
```

5E BASIC COMMANDS-HPL EQUIVALENT

```
CONT - cont
CONT TO - no equivalent
DELETE ALL - erase
DELETE <line number> -del <line number>
EDIT - fetch <line number>
LIST - list
OLD - ldf
REN - no equivalent
RUN -run
RUN"file name" - ldp
SAVE - rcf
SAVEL - no equivalent
STEP - STEP
```

BASIC STATEMENT-HPL STATEMENT

```
COM - not necessary
CLOSE - files or asgn (disk rom)
DATA - not available
DEF - function subprogram, e.g., 'STD'(A,B,C) ret X
DIM -dim
END -end
EXIT - no equivalent
FOR -for
GOTO - gto
ON-GOTO -jmp (close equivalent)
GOSUB - gsb
ON-GOSUB -no equivalent
IF THEN - if <      >;
IF  THEN  ELSE -no equivalent
IF GOTO - if <     >; gto
INPUT -ent or enp
KILL -no equivalent for cassette
     - Kill (disk)
LET - not used
NEXT - next
ON CTRL - rdb(0) or special function key
OFF CTRL - automatic
ON ERROR - on err (ext. I/O ROM)
ON KEY - rdb(0) or special function key
              or on key (systems ROM)
OPEN - files (disk ROM)
PRINT -prt or (sprt or rprt - disk ROM)
PRINT USING -fmt (Gen. I/O ROM)
RANDOMIZE - no equivalent
READ -not available
REM or ! - "<label": at beginning of line
RESTORE - not available
RESUME - iret (Ext. I/O ROM)
RETURN - ret
STOP - stp
STOP ON - not available
TRACE ON - trc
TRACE OFF -normal
```

BASIC FUNCTIONS-HPL FUNCTIONS

```
Y=ABS(X) - abs(X)--->Y
Y=ATN(X) - atn(X)--->Y
Y=COS(X) - cos(X)--->Y
Y=EXP(X) - exp(X)--->Y
Y=INT(X) - int(X)--->Y
Y=LN(X) -  /n(X)--->Y
Y=LOG(X) - log(X)--->Y
Y=PI -     π--->Y
Y=RND -    rnd(x)--->Y
Y=SGN(X) - sgn(x)--->Y
Y=SIN(X) - sin(x)--->Y
Y-SQR(X) - √(x)--->Y
Y=TAN(X) - tan(x)--->Y
Y=ATN(SIN(X)/SQR(1-(SINX)^2))-asn(x)--->Y
Y=ATN(SQR(1-(COS(X))^2)/COS(X))-acs(x)--->Y
```

BASIC STRING FUNCTIONS-HPL STRING FUNCTIONS

```
Y%=ASCII (A$) -          num(A$)--->Y
Y$=CHR$(X%) -            char(X)--->B$
Y$=LEFT(A$,N%)-          A$[1,N]--->Y$
Y$=RIGHT(A$,N%)-         A$[N,len(A$)]--->Y$
Y$=MID(A$,N1%,N2%) -(A$[N,E])--->Y$
                      N2%
Y%=LEN(A$) -             len (A$)--->Y
Y%=INSTR(N1%,A$,B$) pos(A$[N1],B$)--->Y
Y$=SPACE(N%) -            " "--->Y$[1,N]
Y$=TAB(N%) -             not available
Y$=NUM$(N) -             str(N)--->Y$
Y=VAL(A$) -              val(A$)--->Y
Y$=CPOS(Y, X) -          not available
    /   /   \
cursor  row  column
position
```

IEEE-488 COMMANDS-HPL EXTENTED I/O COMMANDS

```
INIT - cli
REMOTE - rem
LOCAL -lcl
CLEAR - clr
LOCKOUT - llo
TRIG - trg
CONFIG @ < > to <device> - polo
CONFIG @ < > - polu
INPUT - red
PRINT - wrt
RBIN -rdb
RBYTE - rdb
WBIN -wtb
WBYTE -wbt
INPUT WBYTE - not available
ON SRQ -rqs or oni with eir
ON PPOL -pol
OFF -automatic
WAIT - wait
```

WAIT FOR – not available

BASIC SYSTEM VARIABLES-HPL SYSTEM VARIABLES

```
ERR - ern
ERL - erl
KEY - rdb (0)
FLEN - return variable for idf statement (tape)
MEM - not available (displayed after listing pgm.)
DATE$ - not available
TIME$ - requires additional ROM
TIME - requires additional ROM
```

BASIC IEEE-488 FUNCTIONS-HPL IEEE-488 FUNCTIONS

```
Y%=SPL(N) - rds (N) ---> Y
Y%=PPL(N) - pol (N) ---> Y
TIMEOUT - time
TERM - not available
```

5F•1 DEGREES VS RADIANS

d = diameter r = radius

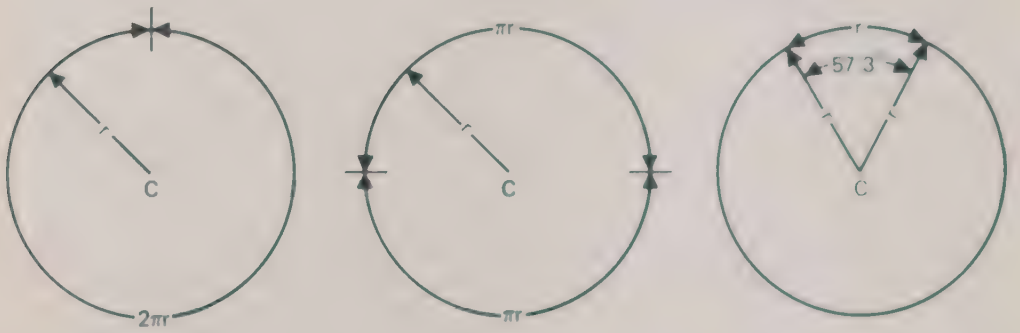
circumference = πd $\pi d = 2\pi r = 6.28r$

$\therefore$ a line 6.28r in length could be formed into a circle with radius r

Since there are 360 degrees in a circle, and it is 6.28r or $2\pi r$ around the same circle, it follows that

$$\frac{360^\circ}{6.28r} = \frac{360^\circ}{2\pi r} = \frac{57.29578^\circ}{r}$$

This angle has been defined as one radian(1 radian = 57.2957 degrees)



1 radian = 57.29578 degrees
1 degree = 0.01745329 radians

$$\text{radians} = \frac{\text{degrees}}{57.29578}$$

$$\text{degrees} = \frac{\text{radians}}{0.017455329}$$

or

degrees = radians*57.29578 radians = degrees *0.01745329

examples: 180 degrees = (0.01745329)(180) = 3.14159 radians
= π radians

$$\frac{\pi}{2} \text{ radians} = (57.29578)\left(\frac{\pi}{2}\right) = 90 \text{ degrees}$$

Common Angles:

2π radians = 360°
π radians = 180°
$\frac{\pi}{2}$ radians = 90°
$\frac{2\pi}{6}$ radians = 60°
$\frac{\pi}{4}$ radians = 45°
$\frac{\pi}{6}$ radians = 30°

5F.2 Grads VS Radians

90 degrees = $\pi/2$ radians = 100 grads

= 1.5707963 radians

or 0.01570796 radians = 1 grad

or 1 radian = 63.661977 grads

$$\text{grads} = \frac{\text{radians}}{0.01570796}$$

$$\text{radians} = \frac{\text{grads}}{63.661977}$$

or

radians = grads * 0.01570796

grads = radians * 63.661977

5F.3 ArcSin and ArcCos: Defined Functions

x = the unknown angle

1st Identity: $(\text{Sin}(x))^2 + (\text{Cos}(x))^2 = 1$

2nd Identity:

$$\text{Tan}(x) = \frac{\text{Sin}(x)}{\text{Cos}(x)}$$

Solving the 1st Identity for Sin(x) and Cos(x)

$$\text{Sin}(x) = \sqrt{1-(\text{Cos}(x))^2}$$

$$\text{Cos}(x) = \sqrt{1-(\text{Sin}(x))^2}$$

Substituting these values into the 2nd identity

$$Tan(x) = \frac{Sin(x)}{\sqrt{1-(\text{Sin}(x))^2}}$$

$$\text{Tan}(x) = \frac{\sqrt{1-(\text{Cos}(x))^2}}{\text{Cos}(x)}$$

It is obvious that ArcSin(x) = ArcCos(x) = ArcTan(x)

It follows that

Equation 1: $\text{ArcSin}(x) = ArcTan \frac{Sin(x)}{\sqrt{1-(Sin(x))^2}}$

Equation 2: $\text{ArcCos}(x) = ArcTan \frac{\sqrt{1-(Cos(x))^2}}{Cos(x)}$

Let Y =Sin(x) and z = Cos(x)

Substituting these values into equations 1 and 2

$$\text{ArcSin}(x) = \text{ArcTan}\frac{Y}{\sqrt{1-Y^2}} \quad \text{and ArcCos}(x) = \text{ArcTan}\frac{\sqrt{1-Z^2}}{Z}$$

and it follows that defined functions can be set up as follows:

```
DEF FNS (Y) = ATN (Y/SOR (1-Y^2))
      \
       ArcSin(x)

DEF FNC(Z)  = ATN (SQR(1-Z^2)/Z)
     \
      ArcCos(x)
```

5G IEEE-488 Decimal equivalents for WBYTE/RBYTE commands

EOI	256	End Or Identify
ATN	512	Attention
DCL	532	Device Clear
GET	520	Group Execute Trigger
GTL	513	Go to Local
UNL	575	Unlisten
UNT	607	Untalk
MLA	544 + Device address	My Listen Address
MTA	576 + Device address	My Talk Address
MSA	608 + Device address	My Secondary Address

Device Address	send MLA	send MTA	send MSA
0	544	576	608
1	545	577	609
2	546	578	610
3	547	579	611
4	548	580	612
5	549	581	613
6	550	582	614
7	551	583	615
8	552	584	616
9	553	585	617
10	554	586	618
11	555	587	619
12	556	588	620
13	557	589	621
14	558	590	622
15	559	591	623
16	560	592	624
17	561	593	625
18	562	594	626
19	563	595	627
20	564	596	628
21	565	597	629
22	566	598	630
23	567	599	631
24	568	600	632
25	569	601	633
26	570	602	634
27	571	603	635
28	572	604	636
29	573	605	637
30	574	606	638

0.	NULL	31.	~	62.	>	93.	]	124.	
1.	B	32.	SPC	63.	?	94.	^	125.	}
2.		33.	!	64.	@	95.	-	126.	~
3.	&	34.	"	65.	A	96.	`	127.	
4.		35.	#	66.	B	97.	a		
5.		36.	$	67.	C	98.	b		
6.	n	37.	%	68.	D	99.	c		
7.	BEL	38.	&	69.	E	100.	d		
8.		39.	'	70.	F	101.	e		
9.		40.	(	71.	G	102.	f		
10.	LF	41.	)	72.	H	103.	g		
11.		42.	*	73.	I	104.	h		
12.	FF	43.	+	74.	J	105.	i		
13.	CR	44.	,	75.	K	106.	j		
14.	<>	45.	-	76.	L	107.	k		
15.		46.	.	77.	M	108.	l		
16.		47.	/	78.	N	109.	m		
17.		48.	0	79.	O	110.	n		
18.		49.	1	80.	P	111.	o		
19.		50.	2	81.	Q	112.	p		
20.		51.	3	82.	R	113.	q		
21.		52.	4	83.	S	114.	r		
22.		53.	5	84.	T	115.	s		
23.		54.	6	85.	U	116.	t		
24.		55.	7	86.	V	117.	u		
25.		56.	8	87.	W	118.	v		
26.		57.	9	88.	X	119.	w		
27.	ESC	58.	:	89.	Y	120.	x		
28.		59.	;	90.	Z	121.	y		
29.		60.	<	91.	[	122.	z		
30.		61.	=	92.	\	123.	{		